INSTANCE-BASED SOURCE CAMERA IDENTIFICATION FOR SMALL SIZE IMAGES USING DEEP LEARNING-BASED TECHNIQUES

ANIMASAHUN IDOWU OLUWASEUN

MPhil

The Hong Kong Polytechnic University

2020

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

INSTANCE-BASED SOURCE CAMERA IDENTIFICATION FOR SMALL SIZE IMAGES USING DEEP LEARNING-BASED TECHNIQUES

ANIMASAHUN Idowu Oluwaseun

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

June, 2018

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

____

(Signed)

ANIMASAHUN Idowu Oluwaseun (Name of Student)

Dedication

This research work is dedicated to Jesus, the author and finisher of my faith for His divine inspiration, wisdom and strength. Also, this work is dedicated to my Mother; Mrs. Grace Kehinde Animasahun for her constant prayers, weekly calls and relentless encouragement, which keep my hope alive despite all the challenges I faced in the course of the programme. You are a great mother and my love for you is unquantifiable. May God keep you strong, healthy and you will live long and reap the rewards of your labour.

Abstract

Source camera identification (SCI) is an area of forensic science that has to do with attributing a photo to the camera that has captured it. SCI has been widely researched using several approaches especially the use of image photo-response non-uniformity (PRNU) fingerprints with normalized correlation and peak to energy correlation as decision parameters. Several classifiers, such as support vector machines (SVM) and neural network (NN), have been used for source camera identification. In some research works, deep learning methods, such as convolutional neural networks (CNN), have been used for camera identification. However, most of the proposed methods have considerably good identification accuracy for identifying the camera models, but poor identification accuracy for individual instance-based SCI. Furthermore, existing source camera identification algorithms mostly have good performance for images of size higher than 256 × 256. Motivated by the knowledge gap in the literature, in the thesis, we propose methods for robust deep learning method, so as to achieve high discriminative power for instance-based SCI for small-sized images. The small-sized images can be due to low resolution or a cropped image patch from an image. Moreover, most of the deep learning-based camera identification methods use images directly as input into the deep networks. However, the contents of the images suppress the camera features, and this has a negative impact on the identification accuracies of cameras. Therefore, we propose the use of noise residues or individual PRNU images so as to suppress the contamination of camera features by image contents. The proposed noise residues are also pre-processed by zero-meaning so as to remove linear patterns, and as a normalization technique for our input data. The work is useful in applications, such as splicing translocations and small-sized forgery detection.

Firstly, we proposed a stacked sparse autoencoder (SSAE) for SCI. Autoencoder is an auto-associative architecture, which is considered suitable for learning input data that is not completely random. Since PRNU is Gaussian distributed and is not completely random, so the autoencoder implemented can learn some interesting structure from the pre-processed noised residues of cameras. The robust features of camera characteristics are learned through stacking several encoding layers of autoencoders recursively. These robust features are then taken as inputs to a regularised softmax classifier for probabilistic predictions of the source camera. We investigated the structure of the SSAE and the hyper-parameters that give optimal performance on our data. For all our proposed deep learning methods, the cross-entropy loss function was used. Furthermore, mini-batch stochastic optimisation was used for updating the network weights. Experimental results on 20 cameras from the Dresden database show that the proposed method achieves comparable identification accuracy when compared with some state-of-the-art methods. The proposed SSAE also generalizes well using the same hyper-parameters on different cameras sets.

Secondly, we propose a robust deep CNN architecture for instance-based SCI. The proposed neural network consists of three convolutional layers and two fully connected layers. The convolutional layer of the proposed CNN includes processing operations, such as convolution, strides, batch normalization, and leaky rectilinear activation (Leaky ReLU). Strided convolution was used as the downsampling operation, instead of the max-pooling in our proposed method. This is because maxpooling aggressively downsamples feature maps, and the quality of the PRNU signal is dependent on the total number of pixel values. Therefore, it will affect the quality of the feature maps. Dropout layers are also used in the fully connected layers to prevent network overfitting. Our dataset consists of a cameras with an unequal number of images (an unbalanced dataset), so we introduced the use of a class weight to the training function. Class weights penalize under or over-represented classes during training. To further prevent overfitting of deep networks and reduce unnecessary computation during training, we adopted the use of early stopping. The output of the FC2 is given as input into a regularized softmax classifier (CNN-SC) for probabilistic prediction of camera classes. Furthermore, after training the proposed CNN, the flattened output of the third convolutional layer with a linear activation was extracted and used as the embedded layer for one-vs- rest linear support vector machines (CNN-SVM). Using one-vs-rest linear SVM classifier gives room for more training samples in a training set for each phase of training. Furthermore, the proposed deep CNN model was also pre-trained on 10 non-target camera classes and fine-tuned on 10 target camera classes. A comparative study with some state-of-the-art methods was carried out. Experimental results show that the identification accuracies of our proposed CNN-based methods (CNN-SC and CNN-SVM) are 18%-25.6% and 20.37%-25.02% higher than four other PRNU-based SCI methods, without and with fine-tuning on a deep CNN pretrained model. We also compared our method with a deep learning-based method, namely content-adaptive fusion networks (CA-FRN). Our proposed CNN-based methods (CNN-SVM without and with fine-tuning) have the identification accuracy 6.12-10.02% lower than CA-FRN for camera brand identification, but have identification accuracy 1.34% and 11.02-12.83% higher than CA-FRN for camera model identification and camera device identification, respectively.

Furthermore, we have evaluated the effectiveness of our proposed deep CNN, fine-tuned on non-target cameras, under geometric distortions, such as JPEG compression with quality factors of 95, 90 and 80, before the extraction of the noise residues of images. The average identification accuracies with post JPEG compression on targeted camera classes are 0.68%, 1.74%, and 3.37% lower than the average identification accuracy, without post JPEG compression, for the quality factors of 95, 90 and, 80 respectively. This shows that our proposed CNN-system, with fine-tuning is robust to post JPEG compression with only little reduction in accuracy, as compared to those images not being compressed.

Finally, we propose a learning method to extract the PRNU fingerprint and to perform camera identification. The extraction of the PRNU fingerprint of a camera from a smooth, plain image is much easier than from a natural or cluttered image. Based on this observation, we propose both a manual and automatic curriculum learning method for instance-based SCI deep residual CNN (ResNet). Residual connections are added to our proposed deep CNN architecture so as to generate more robust representational bottlenecks, and also to tackle the vanishing gradient problem. The idea of curriculum learning (CL) is to train a system, which may be a student or a deep network, from simple concepts to hard concepts. For the manual CL method, the proposed ResNet is first trained with flat images. Having trained with flat images, those cluttered or natural images are mixed with flat images to continue training up the network. In real applications, all the available images are usually of natural images. Therefore, the last stage of our proposed CL uses natural images only. For the second CL algorithm, the features of training images are extracted from the softmax layer of the trained ResNet, and the cross-entropies of each instance of the extracted features are calculated. The indices of the sorted cross-entropies are used to classify the training images as simple or hard images. Our experimental results show that ResNet-SVM has 2.18% and 0.27% higher identification accuracies than CNN-SVM, without and with fine-tuning respectively. For the manual CL, our experimental results show that ResNet-SVM has 3.74% identification accuracy higher than training with no curriculum learning. Furthermore, our proposed automatic CL approach only has 0.47% identification accuracy higher than training with no CL. In conclusion, our proposed deep learning methods for instance-based SCI can still achieve good performance using a small data size, unlike a large amount of data required for good performance in some camera identification problems. Moreso, unlike the inability of the proposed CNN-based method in a work published in 2019 to acheive better identification than a PRNU-based technique, our work can acheive better identication accuracy than the conventional state-of-the-art methods that use PRNU-based methods using identical settings.

List of Publications

I.O. Animasahun, and Kin-Man Lam, "Deep Residual Convolutional Neural Network with Curriculum Learning for Source Camera Identification," in *International Workshop on Advanced Image Technology*, accepted for publication.

Acknowledgments

I would like to sincerely express my appreciation to all the supervisors involved in my research programme supervision. Firstly, I will like to appreciate Prof. Kenneth K.M. Lam for his agreement to take up the supervision of my programme as the Chief Supervisor during the revision of my thesis for resubmission for examination process. I also want to express my profound gratitude for all you taught me and for the experience have gained under your supervision. Thanks so much for your all your support. I appreciate all your contributions.

Also, I would like to express my appreciation to Dr. LAW Ngai-Fong, Bonnie who supervised my research programme prior to its continuation by a three-member supervisory team. Thanks for your supervision and all I learned under your guidance. Moreso, I would like to express my appreciation to other co-supervisors: Prof. SIU Wan-Chi, and Dr. LEUNG Hung-Fat, Frank for their contributions.

My sincere and profound gratitude goes to the supervisory team: Dr. MAK Man-Wai, Dr. CHAN Yuk-Hee, Chris and Dr. CHAN Yui-Lam for all your support during the transient period of your involvement in my supervision. I sincerely appreciate all your efforts including giving instructions, corrections, suggestions, checking of program codes and the use of your time despite your busy schedules. Thanks so much Sirs and am highly grateful for all your support.

I will like to also express my profound gratitude to the Dean of Faculty of Engineering; Prof. Hau Chung Man, and Prof. WAI Ping-Kong, Alexander, the Vice President (Research Development) for their quick intervention on the administrative issues as regards my programme. Thanks to Dr. Cheng Virginia (RO) and Ms Irene Ho (RO) for their moral and administrative support. I also want to appreciate all the other members of Research Committee on the resolution regarding issues on my programme. Thanks for your administrative involvement.

I thank my fellow laboratory mates for the stimulating discussions and the insightful research group presentations we had together. A sincere thanks to Dr. Masebinu Samson, highly beloved friend, for his moral support and diligent proofreading of my thesis. I would also like to thank Dr. Tayo Abolude, Dr. Wole Soyinka, Dr. James Amuda, Dr. Loto Oluwasayo, Dr. Boluwatife Abidoye, and all other friends for your moral support, guidance and continuous encouragement to strive towards my goal.

Last but not the least, I would like to thank my family: my parents, my brothers and sister for supporting me morally and prayerfully throughout my programme.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AE	Autoencoder
ANN	Artificial Neural Network
BGD	Batch Gradient Descent
CBI	Camera Brand Identification
CDI	Camera Device Identification
CE	Cross-Entropy
CL	Curriculum learning
CMI	Camera Model Identification
CNN	Convolutional Neural Network
CNN-SC	CNN with Softmax Classifier
CNN-SVM	CNN with One-Vs-Rest Linear SVM Classifier
CNN-SVM1	CNN-SVM without fine-tuning
CNN-SVM2	CNN-SVM with fine-tuning
DBN	Deep Belief Networks
DNN	Deep Neural Network
EXP	Exponential Cost Functions
GPU	Graphic Processing Units
MAP	Maximum A Posterior

MBSGD	Mini-Batch Stochastic Gradient Descent
MLE	Maximum Likelihood Estimation
NN	Neural Network
PCA	Principal Component Analysis
PReLU	Parameterized Rectilinear Unit
PRNU	Photo Response Non–Uniformity
ReLU	Rectlinear Unit
ResNet	Dep residual Neural Network
RBM	Restricted Boltzmann Machines
SAE	Stacked Autoencoder
SCI	Source Camera Identification
SGD	Stochastic Gradient Descent
SM	Softmax Classifier
SPN	Sensor Pattern Noise
SSAE	Stacked Sparse Autoencoder
SSE	Sum of Squared Errors
SVM	Support Vector Machine

Chapter 1 :

Introduction

1.1 Background

Manipulation of images is not only peculiar to digital photography but also to silver-based photography or analog photography. Images from both digital and analog photography can be tampered or alterd. Digital imaging technology is still considered in forensic science has been better than analog photography in tracing the source of an image. Unlike analog images, digital images are usually attributed with additional information which can be useful for or suggest the possible transformation process carried out on the digital images. These peculiar properties have made digital imaging technology an important area of application in forensic analysis of images. Digital images consist of pixels or picture elements. The pixel is an estimation of a colour. A digital image can be a true colour image with three channels (red, green and blue) or a grey level image (black and white image) with a single channel. Most of the forensic analysis are done using grey-level transformation and this is due to easy storage and fast computation time. The overall identification accuracy when grey images are used compared to true colour transformation, is insignificant [1]. Images and videos have been widely used as evidence in a court of law to depict the truthfulness of crime-related offenses or illegal actions. The authenticity of the multimedia content provided as a proof is now a question of interest [2]. In film photography, alteration in the output image is limited; since it takes time and also costly. In contrast, images from digital photography can be easily altered due to the rapid development of digital electronic tools. The availability and cost-effectiveness of these tools have further contributed to their widespread of tampered images. These tools can be used to alter the originally captured multimedia content.

A question of interest is that, can the authenticity of the digitally captured images or videos still be guaranteed since they can now be easily adjusted by these processing tools or readily available computer graphics [2, 3]? Another question of interest is whether the images captured are purely synthetic images generated from computer software. One common software with a very high number of users all over the world is the Adobe Photoshop software. The design of this software greatly enables easy manipulation of any desired parts of the original images. The changes in the altered images are most of the times not perceptible to the human eyes. This makes it difficult to know a multimedia content has been altered by mere observation. Therefore, the integrity of digital images is in doubt, since it cannot be guaranteed that they have not been forged [3]. In order to circumvent these problems, digital image forensics researchers investigate some fundamental features of a real image. Apart from the manipulation of the visual content of images, another area of concern is the authenticity of the device that captured the images in question. This is the question of whether the images are from a camera, scanner, and cam decoder or whether they are computer graphic images. Though exchangeable file format of images usually have metadata headers that contain information about the devices that captured the images. However, the information on the metadata headers can be edited or deleted. Therefore, the use of such images in forensic analysis as an evidence in a court of law could be misleading [2]. There are other aspects of digital forensic science such as tampering discovery, steganalysis, recapturing identification, recovering of processing history, computer graphics identification, device temporal forensics and anti-forensic [3]. As a result of the rapid production of digital cameras and scanners, forensic research has mainly focused on camera and scanner source identification [2]. The two categories of techniques used for multimedia forensics are classified into active and passive forensics. Figure 1.1 shows the diagrammatic representation of the classification of the techniques used for multimedia forensics [3].

In Figure 1.1, the active forensics is further divided into two parts; which are the precomputation of the harsh part of the image and also information hiding. Active forensics involves computing beforehand some delicate properties of some severe or bad parts of an image. The second involves encoding the original image with a secret data before it will be sent through a public channel. In the case of alteration in the sent image, the forensic decision is then made by comparing the encoded information of the original image with the extracted encoded data from the manipulated or tampered image [3]. According to [4], during the image production, watermark inclusive of iris biometric data was used as the encoded data. Verification of the authenticity of the image can now be achieved by comparing the extracted watermark of the suspected image with the original image. The techniques usually used in passive forensics, are based on the fundamental idea that the source of an image (camera or scanner) contains sufficient features that can be used to verify its authenticity when compared with its forged image. Active forensics, have the advantage of being highly effective in justifying the authenticity of an image but the added artifacts due to the encoded data, further degrade the quality of the output image. Its strict requirements in ensuring accuracy is also a limitation. Most of the forensic researchers adopt the passive forensic techniques since they do not impose specific requirements or additional artifacts apart from the one due to camera-in processing techniques [4]. The focus of our work is on the source camera identification. There are two types of camera identification problems, namely camera model identification, and instance-based source camera identification (SCI). The former attempts to identify only the model/brand of the camera that was used to capture a photo while the latter attempt to identify the individual or source camera [5, 6].

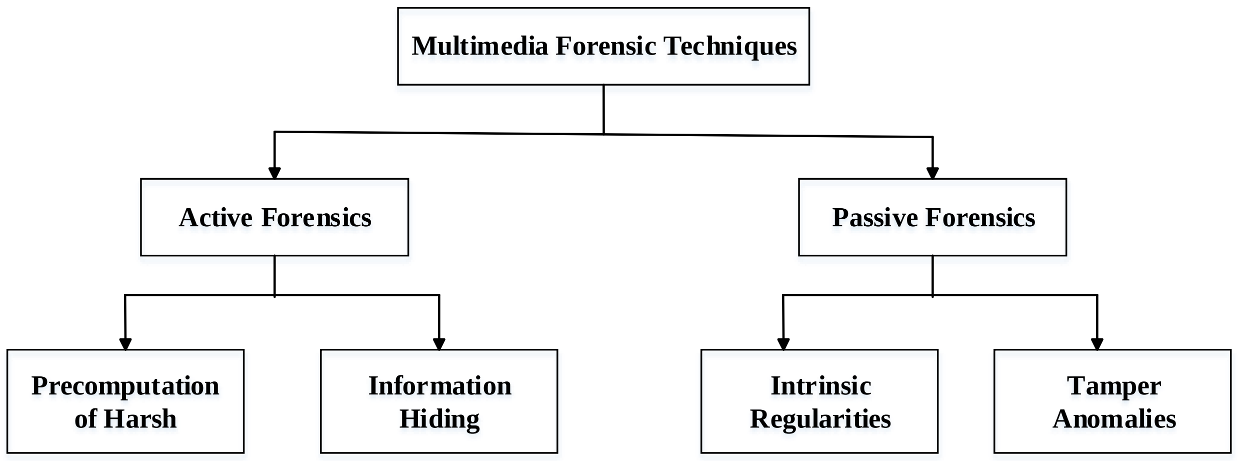


Figure 1.1. Classification of the Techniques Used For Multimedia Forensics [3].

1.2 Motivation

The need to know the source of an image is that it helps to provide some useful pre-information necessary for further analysis on the image. For example, in a situation where the source of a captured image is known, it could help in locating the photographer and also the device. The knowledge of the model of a device helps to further acquire more information about the device by getting the model specifications and features from several resources available online. Recently, the sensor noise defects of a camera have been widely explored and commonly used as camera fingerprints. The sensor noise defects include readout noise, shot noise, fixed pattern noise, and the photo response non–uniformity (PRNU). The PRNU has been the most commonly used camera fingerprint in the literature [7-16] due to the universality, dimensionality, and robustness. The PRNU is also called the sensor pattern noise (SPN) of a camera. Furthermore, several classifiers such as support vector machine (SVM) and neural network (NN) have been used to solve source camera identification problem [17, 18]. Some of the literature that has achieved very high camera identification accuracy used a combination of several features like 34 features used in [18]. Despite all the several approaches used in literature [7-16], there are still problems that need to be solved. Is it still possible to achieve higher accuracy just with a single feature? Can higher accuracy still be achieved at lower image resolutions irrespective of the texture complexity of images? Can fusion of classifiers help in achieving better accuracy or is there a better classifier that could help further increase the classification accuracy using only PRNU as the input feature? Furthermore, the work in [19], carried out a comparative study on the proposed CNN-based SCI and on a PRNU-based method using identical settings, concluded that the PRNU-based method has higher identification accuracy, at a lower computational cost, than the proposed CNN-based method. An interesting question is whether a more accurate deep network architecture can be designed for instance-based SCI, by using effective training algorithms. Another class of machine learning that has high discriminating power and is capable of achieving robust feature representation is deep learning. This method has been widely used in many applications such as face recognition [20], image classification [21], fingerprint liveness detection [22], automatic malware signature generation and classification [23], modulation format identification in coherent receivers [24] and classification of hyperspectral images [25]. The ability of a deep neural network (DNN) to automatically learn features is what makes it a suitable technique for many applications. Deep learning is a hierarchical learning. It is a subset of machine learning which uses algorithms to self-learn robust representations of a data when given recursively through several NN layers. Several supervised and unsupervised deep learning modules have been explored over the years. These include restricted Boltzmann machines (RBM), deep belief networks (DBN), convolutional neural network (CNN), autoencoder (AE), denoising autoencoder [21, 22, 26] and residual convolutional neural network [27]. Due to the current success of deep learning in achieving invariant feature representation and high discriminative power, the aim of the research is to apply deep learning techniques for source camera identification

Also, existing source camera identification algorithms mostly have good performance for images size ranging from 256×256 and above [7-16]. Motivated by the knowledge gap in the literature, in the thesis, we carefully designed robust deep learning methods with high discriminative power for instance-based source camera identification for small-sized images. The small-sized images could be due to low resolution or a cropped image patch from an image. In our work, small sized images are defined as images with sizes from or below 256×256. For the proposed SSAE, we carried out experiments using 256×256, 128×128 and 64×64 image sizes while for CNN-based methods, we used 64×64 image size. Also, some deep learning-based camera identification methods used the images directly as input into their proposed deep networks [28-30]. However, the contents of the images suppress the camera features, and this has a negative impact on the identification accuracies of cameras especially for instance-based source camera identification. Therefore, we proposed the use of noise residues or individual PRNU images so as to suppress contamination of camera features by image contents. The proposed noise residues are also pre-processed by zero-meaning so as to remove linear patterns and as normalization technique for our input data. The ultimate goal is to increase the camera identification accuracy for small size images compared to some existing methods in the literature. The work is useful in applications such as splicing translocations and small-sized forgery detection. Therefore, in this work, we propose basically three deep learning-based methods for instance-based source camera identification and the motivation for each method are discussed as follows.

Firstly, we propose stacked sparse autoencoder for source camera identification. An AE is an artificial neural network with one hidden layer. It helps to learn a compressed representation of an input data. It does this by first transforming the input data to hidden features (encoding) and these features are then mapped back to obtain an approximation to the input data (decoding) [31]. When sparsity constraint is imposed on the hidden layer of an AE, it is called sparse autoencoder [21]. The constraint is added to ensure that a good representation is obtained even when the number of hidden units is large. Stacking the encoding features of hidden layers of autoencoders recursively is called stacked AE (SAE). It has been used successfully for classification problem [20, 32, 33]. The work in [21] used a deep learning approach called stacked sparse autoencoder (SSAE) with softmax classifier (SM) for image classification. Their experimental results show that, only 60 samples per class were used for pre-training and 12 samples per class for validation. Despite the small training size, an overall accuracy of 91.67% was achieved. This demonstrated that a small set of input data but with high quality can still produce a good result. In [34], for an input data to be considered suitable for an autoencoder, it should not be completely random. The input should not be identically and independently Gaussian. For an input to be identically and independently Gaussian means each of its random variables must not have the same probability distribution as the others and all must be mutually independent. It can also be suitable for a dataset that has at least some structure. Some of the inputs features in the data should be correlated. Although PRNU has a noise-like characteristic there is a correlation between the PRNU of training and testing images belonging to the same camera instance. AE is data specific and is not suitable for learning features directly from images which do not have specific properties [35]. Therefore, for SCI, SAE is considered suitable since there is a correlation between PRNU of images of the same camera instance. The combination of the SSAE and SM can be used to extract a robust and invariant representation of features of the PRNU for source camera identification. The basic concept of SAE is called the layer-wise pre-training through several hidden layers without the knowledge of class levels. The idea is to learn robust features at each layer of the network progressively [32]. These features are called deep features. After the training stage, a supervised learning can then be carried out on the learned features from the pre-training using any multi-class classifiers such as SVM or SM.

Secondly, we investigated the CNN architecture that is best suited for obtaining robust noise features of cameras. CNN are specifically designed for recognition of images and has been widely used in several applications involving image and speech processing, text classification and reinforcement learning for a board game and video game. CNN has also been used in digital forensics applications. For example, Bayer et al. [36] proposed a CNN architecture for the detection of manipulated images. CNN was adopted for detection of median filtering of images [37]. Besides, CNN has also been applied for camera model identification [28, 29, 38]. In-camera model identification, the work in [29] achieved an accuracy of 94% using 27 camera models for 32×32 images sizes. This accuracy was achieved when only one instance of the camera models was used. When multiple cameras of the same model were included, the accuracy dropped to 29.8%. This shows that the CNN architecture in [29] was unable to achieve good performance in instance-based camera identification. A different CNN architecture was proposed for 64×64 images sizes in [28] and over 93% identification accuracy was achieved. However, the proposed technique is for camera model identification, rather than instance-based identification. The background assumption in both [29] and [28] is that CNN is able to learn features about the processing pipelines of cameras directly from the images without the use of hand-crafted features. Since CNN usually requires the use of large dataset for implementation, images were divided into patches for training. Instead of using images as input, PRNU was used as input in [23] for camera model identification. Their experimental results show 98.0%, 97.09% and 91.9% for 12, 14 and 33 camera models respectively on 256×256 image sizes. Despite the good performance in camera model identification, the work in [38] did not consider instance-based source identification. We believe that the structures of the CNN have to be modified so that the CNN can have good performance in instance-based source camera identification. Images contain scene details and camera model features such as color filter array interpolation and lens distortion which are not unique to each camera device. However, PRNU is unique for each individual device. We thus propose to use PRNU images instead of the original images as inputs to CNN. We believe that PRNU images can provide quality input information to the CNN so that the training does not need a large amount of input data. Consider the work in [21] that used a deep learning approach called stacked sparse autoencoder with softmax classifier for image classification. In one of their experimental works, only 60 samples per class were used for pre-training and 12 samples per class for validation. Despite the small training size, an overall accuracy of 91.67% was achieved. This demonstrated that a small set of input data but with high quality can still produce a good result. As earlier stated, existing PRNU-based source camera identification algorithms have good performance for images size larger than 256×256. Identification of small size patch using sensor pattern noise is still challenging for camera identification. This is because the estimated PRNU fingerprint weakens as the number of pixels reduces. However, identification from small image patch is important, especially if forgery regions are to be located. In this work, we will investigate if the CNN approach can achieve reliable identification from small image size such as 64×64. CNN architecture using the sensor noise may not require so very huge data size before it can achieve superior performance as compared to huge data size required for good performance in other applications. This will further reduce computational demand required for the proposed CNN implementation.

Lastly, to train a deep neural network, a large amount of training data is required, in order to avoid overfitting. Therefore, effective training algorithms are important for deep neural networks to achieve good generalization power. For source camera identification, we usually have limited examples from each of the cameras under consideration. Furthermore, the PRNU fingerprints of a camera is difficult to detect from an image, if the image has complicated patterns. In other words, the extraction of the PRNU fingerprint from a smooth, plain image is much easier than from a cluttered image. Based on this observation, we employ curriculum learning to train a deep ResNet for source camera identification. The idea of curriculum learning (CL) is to train a system, which may be a student or a deep network, from simple concepts to hard concepts. This learning approach allows the system to train up from handling simple tasks too hard tasks. The use of curriculum learning can help to improve the speed of global convergence during training and a better local minimum can be achieved [39].

1.3. Organization of the Thesis and Contributions

In this thesis, we focus on the use of deep learning techniques for instance-based SCI for small image sizes. This will be achieved using three deep-learning based proposed methods. The learning will be done on the noise residues of cameras. Apart from the Chapter 1 which focused on the background and motivation for the work and proposed methods, there are other six chapters in the thesis.

In Chapter 2, we reviewed SCI, image features usually used for SCI and a review of some of the state of the art techniques that are well used for the extraction of the PRNU of cameras.

Finally, we carried out a comparative analysis on some of the states of the art methods so as to validate the effect of the size of the image on identification accuracy. The experimental results show that the lower the image size, the lower the identification accuracies of cameras. This shows that the noise residue becomes weaker as the image size gets smaller.

In Chapter 3, we reviewed the historical background of the artificial neural network, introduction to deep learning, objective functions and supervised optimization algorithms. The aim of the chapter is to understand the working principles of some of the methods that will be applied in Chapters 4 and 5.

In Chapter 4, we introduce the use of SSAE for instance-based SCI. We did a review on sparse and denoising autoencoders. We introduce the network architecture for the proposed SSAE and also the PRNU preparation as input into the SSAE. The trained features using SSAE are then given as input to a regularized softmax loss layer for probabilistic prediction of source cameras. We investigated the structure of the SSAE and hyper-parameters that give optimal performance on our data. Experimental results show that significant overall identification performance comparable with some existing methods on the Dresden database and better performance on our own dataset when compared with some state-of-the-art methods for all tested image sizes. Also, the proposed network also generalizes well using the same hyper-parameters on different cameras set.

In Chapter 5, we introduce our proposed deep CNN for instanced based source camera identification of small-sized images. We discussed the working principle and the training process of CNN. Also, we discussed the proposed CNN architecture, its training, fine-tuning process and selection of hyperparameters. Apart from building CNN from the scratch for SCI, we also proposed the use of transfer learning using the proposed deep CNN as the pre-trained model on non-target camera classes and the classification was carried out on target camera classes and this approach has improved performance than without fine-tuning on pretrained models on non-target camera classes. We proposed, a carefully designed CNN architecture for instance source camera identification by investigating suitable CNN-based architecture to extract robust features of camera noise residues rather than extracting features directly from the camera images as done in Bondi et al. [28]. Since we are using the noise residues of cameras, we investigated suitable normalization technique, CNN signal processing operations, depth of CNN architecture and effective training algorithms. Strided convolution was used as the down-sampling operation instead of the max-pooling in our proposed method. This is because, max-pooling aggressively down sample feature maps and the quality of the PRNU signal is dependent on the total number of pixel values and hence, it will affect the quality of the feature maps.

For the training methods, apart from the use of mini-batch stochastic gradient descent and categorical cross-entropy loss, sparsity constraint and weight regularization methods are used to further prevent model overfitting. Finally, we introduce the use of class weights to the training function. Class weights penalize under or over-represented classes in the training set. Our proposed class weight function is passed as a dictionary into the class weight parameter of the network training function. Experimental results on cameras from the Dresden database show that our proposed deep methods using a single image patch (64×64) from each image achieve superior performance than the compared methods using a small data size, unlike the requirement of using a huge dataset used for CNN training for some camera identification problems. This further reduces computational demand required for the CNN training and can be attributed to the fact that our proposed network was learned on preprocessed noise residues less contaminated by the image scenes and hence, can extract high discriminative features for instance-based camera identification.

Moreso, we carried out experimental evaluation on the robustness of our proposed deep CNN methods to geometric distortion such as JPEG compression. There is positive influence on the quality of the features maps of the target cameras based on the weights of all the layers of the pre-trained CNN being updated during training. Therefore, experimental results show that the effect of post JPEG compression on the quality of the PRNU fingerprints of cameras can be suppressed by fine-tuning on a pre-trained model for data with related probabilistic distribution.

In Chapter 6, we introduce the concept of residual convolutional neural network and also the background philosophy behind the use of curriculum learning in deep neural networks. The architecture of the proposed ResNet is discussed and it basically consists of additional residual connections to our proposed deep CNN architecture in Chapter 5. Also, we set out the procedures for both our proposed manual and automatic algorithms. Finally, we evaluated the proposed curriculum learning on 10 cameras from the Dresden database and the experimental results show the benefits of adding residual connections to our initially proposed deep CNN without residual connections. Also, our experimental results show that training on easy training examples before hard examples (manual CL algorithm) can contribute to increasing identification accuracy of the proposed ResNet model.

Chapter 7 concludes the thesis and also gives the future research directions.

Chapter 2 :

Review of Source Camera Identification

In this section, the processes involved in the formation of images using a digital camera were discussed. Also, several camera features that have been explored for SCI are discussed. Finally, we compared some state-of-the-art methods for different image sizes as to observe the effect of identification accuracy on different sizes of images. The motive of the comparison is to check the effect of different image sizes on the camera identification accuracy.

2.1 Formation of an image in a Digital Camera

The block diagram in Figure 2.1 gives the signal processing stages involved in the formation of an image in a high-end digital camera. It can be observed from Figure 2.1 that the light from the multimedia content passes through some set of lenses. The light from the lenses goes through the optical filter. The basic function of an optical filter is to improve the image. It helps to control the brightness levels of the image by reflecting its actual wavelengths. The optical filters commonly used are anti-aliasing filters and infrared filters[5]. Each pixel of an image has a colour, therefore, photons of the light from the optical filters are passed through an array of filters called colour filter array. These photons are converted to electrons by the imaging sensor. There are different imaging sensors. These are charge coupled device, complementary metal-oxide semiconductors, junction field effect transistors and CMOS-Foveon X3 [5]. The differences in the aforementioned types of camera sensors are based on the kind of semiconductors used. Induced voltages or electrons are produced at the sensors. They are in analog form and hence an analog to digital converter unit of the digital camera helps to convert the electrons into digital values. The outputs of the sensor are digital negatives or raw images.

For these raw images to be in a viewable format, software for raw image converters is used in case the camera is set to output raw images. If the camera is not set to raw format, then the output of the sensor is further debayered using demosaicing algorithms to restore its full colour and help depict the real world scene. Due to artifacts incurred during the processing of the image, some post-processing techniques such as white balancing, gamma correction and compression are further applied to the demosaiced image[40]. Finally, the enhanced image output is then stored in a camera memory device depending on the desired format [40].

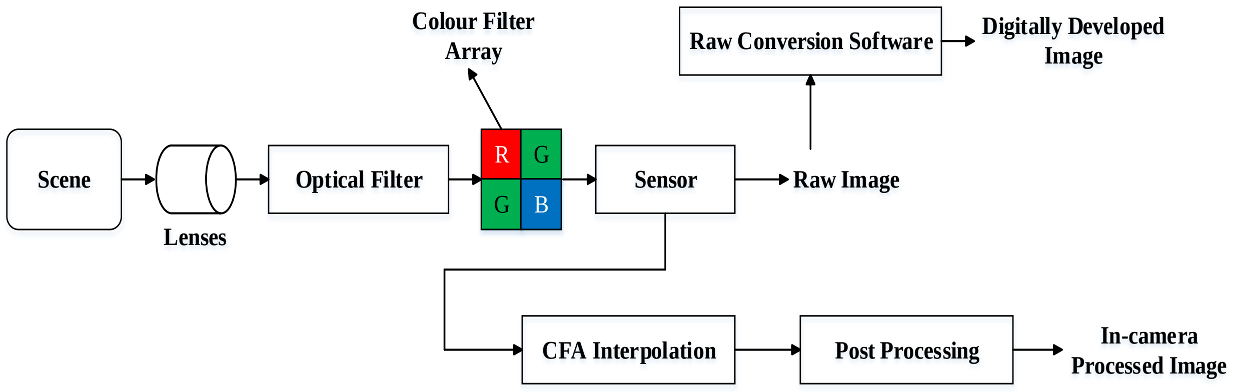


Figure 2.1. Processing stages of a high-end digital camera [34].

2.2 Digital Camera Features used for Source Camera Identification

As described in Section 1.1, multimedia source identification can be broadly divided into either image source identification or source class identification. For the source class identification, the class can either be a scanner, camera or cam decoder. SCI can be defined as the science of identifying some features of a camera on an image and using these features to identify the specific camera used to capture the image. Some of the image features that have been used in the literature are briefly discussed in this section.

2.2.1 Lens Distortion Features

These are some features in images due to defects from the manufacturing process of the lens. There are different types of lens distortion that have been used in literature and some of them are lens radial distortion, chromatic aberration, spherical aberration, field curvature, astigmatism and so on. Out of all these features, the lens radial distortion appears to be the one with most severe effect on the quality of the image. It is highly perceptible especially in less expensive cameras and it depicts itself as straight lines into curved lines in the captured image. There is first and second order radial distortion. Choi et al. [41] used the second-order radial distortion in their work. The features used are based on two computed parameters with Kharrazi's features reported in [18]. Considering a fixed optical zooming, the work achieved 89% accuracy using five different cameras' models. Variation in refractive indexes of lens materials for different wavelengths also leads to another type of lens distortion called lateral chromatic aberrations. The misalignment in colour channels of the image is caused by the effect of the refractive indexes and this prevents the colours from being properly focused. Van et al.[42], used image registration approach to finding the optimal optical center parameters. Three different cameras' models were used and they achieved 92% identification accuracy based on SVM classifier.

2.2.2 Demosaicing Regularities

Demosaicing is an approach for the reconstruction of full colour images from an image sensing device. Through this process, pixel colours are extracted by interpolation to retrieve the image in a viewable format. In order for the raw images to be in a viewable format, demosaicing or debayering algorithms are used in reconstructing the full-colour images from the imaging sensor.

All the pixels colours individually extracted using colour filter arrays are then interpolated or reconstructed to get back the image in it's viewable and full colour format using the demosaicing algorithms. The demosaicing algorithms vary from one camera model to the other. The algorithms are camera specific and differs greatly[3]. Multimedia forensics, adopt these differences as the means of uniquely identifying one camera model from the other. As reported in Popescu and Farid [43], an accuracy of 97% has been obtained by adopting the expectation maximization algorithm. This algorithm uses the MAP by estimating the interpolation noise variance. The work done by [44], has been further extended by constructing a SVM for smooth and non-smooth regions of the output image. These parameters were obtained by finding the second derivatives of the elements in the rows of the matrix of the output image. Long et al. [6] explored the error features due to the estimation of the interpolation filter coefficients as demosaicing algorithms. The extracted error features were used to train a NN. In the work, not all the error features were used, principal component analysis (PCA) was used to extract the most significant features sufficient for the camera model identification. Using, four camera models, the work achieved an overall accuracy of about 100%. Furthermore, Swamminathan et al. [45] , Cao and Kot [46] have done further works by using enhanced demosaicing features.

2.2.3 Sensor Noise Defects

To begin with, it will be necessary to have the basic understanding of the different sensor noise defects at different levels of the signal transfer. Photon-transfer curve, shown in Figure 2.2, can be used to explain the sensor noise defect. This is demonstrated using the photon transfer curve. This curve is shown in Figure 2.2. This curve is used as the standard steps to be followed or considered during the manufacturing process of a camera. It helps to give the information about the read noise, shut noise, fixed pattern noise (readout noise), full well capacity, fixed pattern noise camera's sensitivity and PRNU [34].

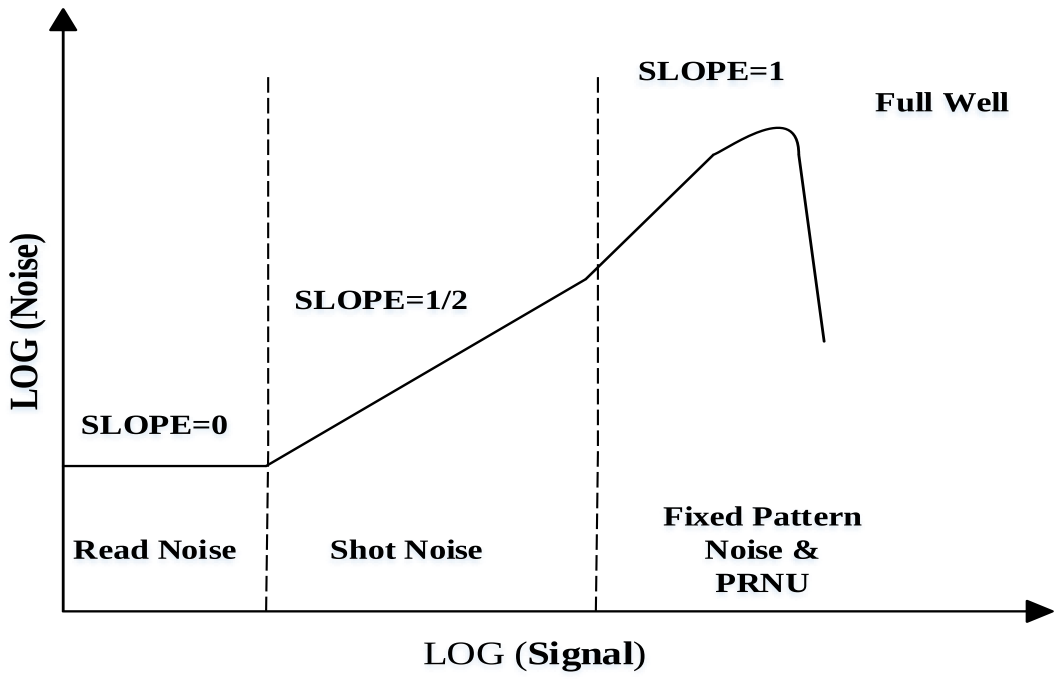


Figure 2.2. Photon Transfer Curve [34].

2.2.3.1 Readout Noise

The first stage of the photon transfer curve is the readout noise. This is the bias frame and it has a slope of zero. The read noise or readout noise is caused by the electronic devices of a digital camera. The voltage induced by the photons at the sensors are further amplified and the amplifiers can only give an estimate of the signal value and this is one of the causes of the readout noise. The readout noise of a digital camera can be measured by calculating the mean, standard deviation of overlapping frames. The set of the calculated means will be plotted against the standard deviations and a scatter plot will be obtained. By using regression, the scatter plot is then converted to a line plot. The intercept of the standard deviation when the mean is zero is then taken as the readout noise of the camera. The readout noise is uniformly distributed across the image. In [40], two experiments were carried out using the readout noise. The first experiment combined two pictures from the same camera having different exposure rates and the second experiment used the readout noise of the images without considering differences in exposure rates. Six cameras were used for the analysis and average cameras' identification accuracies obtained were 88.3% and 79% respectively. For the readout noise to be an effective feature in source camera identification, there is a need for the actual readout of the noise of the camera to be known beforehand [40].

2.2.3.2 Shot Noise

The shot noise as shown in Figure 2.2, is the phase with the slope of 1/2. When the light intensity increases, there will be an increase in the transfer of photons from the camera lens to the sensor. This causes a random movement of the photons and it leads to what is called the shot noise. It has a slope of half because the rate at which photons arrive at the imaging sensor and a number of incoming charges are inversely related [40].

2.2.3.2 Fixed Pattern Noise

Photons can be thermally generated in a camera due to exposure, temperature or the camera settings [40], [47]. They are noises caused by thermal effects. Given that only the fixed pattern noise is assumed to be the only sensor defect, then, the pixel's output is expressed in [47] as,

$$Y = I + \tau D + c \tag{2.1}$$

where I is the input image, D is the dark current, τ is the multiplicative factor and c is the camera offset setting. The photons generated in the camera due to thermal effects are then converted to electrons at the imaging sensor. These free electrons are called dark currents. They are called fixed pattern noise because they maintain their pixel's value across all frames. These currents are weak but its effects can be well pronounced if the images are taken by the night. When D and c are extremely high, it leads to point defects called hot and stuck pixels respectively. These two point defects are used in forensic science for class source classification and digital photograph age estimation [47].

As earlier shown in Figure 2.2 , PRNU is a noise defect with a slope of 1 and it is a pattern noise most common in natural images. Unlike fixed pattern noise, PRNU is resistant to exposure, to humidity and temperature. Imaging sensors are usually made up of silicon waivers. The inconsistencies in the nature of silicon waivers affect the responsiveness of photons of being easily converted to pixels. The rate at which the photons are converted to electrons is called quantum efficiency [40], [47]. The inconsistencies in the pixels' responsiveness lead to a sensor defect called PRNU. This could also be caused due to malfunction in the design of the imaging sensor during the manufacturing process and dust specks. The PRNU, also consists of low-frequency components, such as the effects of light refraction on dust particles. The PRNU component is usually adopted for SCI because it is an intrinsic feature of a digital sensor. For dark images, fixed pattern noise is more pronounced compared to PRNU but PRNU is strongly present in saturated images. The techniques usually used in the manufacturing of image sensors are identical. This makes the PRNU in all the sensors to have related features It has also been found to be dominant in the high-frequency components of an image [5].

2.2.3.3 Photon Response Non-Uniformity

Most of the literature has used PRNU either independently or with other features. It is becoming most commonly used camera fingerprint. There will be a need to further explain why PRNU is widely used for forensic analysis. The first forensic interest of the PRNU is its uniqueness to all sensors in digital cameras since it has more information regarding each sensor. For this reason, no two cameras have the same PRNU fingerprint. It is possible for an image to be recognized by cameras of the same model, but these cameras still have different statistical parameters. Secondly, there is no camera irrespective of the post-processing stages that does not exhibit PRNU. This universality and generality make it a desired fingerprint for forensic analysis. Another very useful characteristics of PRNU is its high stability irrespective of the number of years it has been extracted. Finally, PRNU can withstand the artifacts caused by JPEG compression, white balancing, gamma correction and filtering in camera operation processing. Based on these properties, the camera source of an image can be known once the PRNU of that camera is available. PRNU as camera fingerprint can also be used to know the age of a digital photograph and also for verification of image authenticity [47]. Therefore, PRNU is used as the camera fingerprint for the source camera identification in this research work. An example of an extracted PRNU from a digital camera is shown in Figure 2.3.

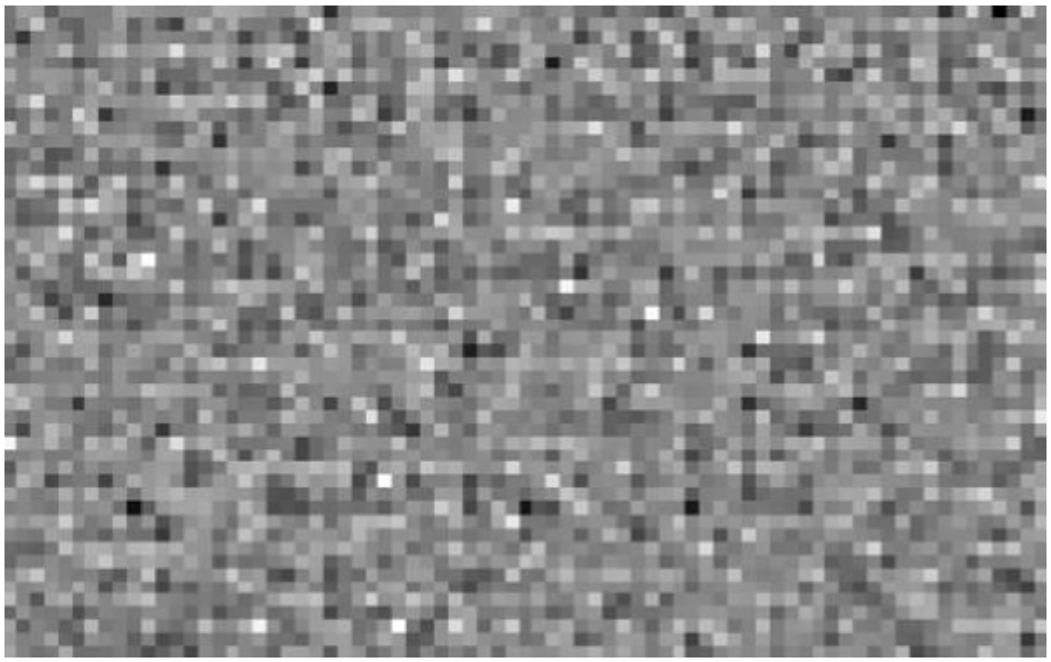


Figure 2.3. Close-up of the PRNU factor K enhanced for visualization [47].

2.3 Extraction of the Camera SPN

The first step before the extraction of the SPN of a digital camera is to know the general pixel - output model. This section focusses on the pixel-output model and also discusses some of the approaches that have been adopted in the extraction of the SPN such as Basic SPN, maximum likelihood estimation (MLE) SPN, SPN using enhancing models, phase SPN and other extraction methods.

2.3.1 The Pixel Output Model

The general pixel output model based on the sensor's imperfections discussed so far, is given in [47] as,

$$Y = I + IK + \tau D + c + \emptyset \tag{2.2}$$

where Y is the noisy image, K is the multiplicative factor, IK is the PRNU noise, $\emptyset$ is the model of all other sensor's imperfections such as readout noise, shut noise, quantization noise and photonic noise. All the noises, classified under $\emptyset$ are not usually used for forensic analysis because of their random nature. The presence of this PRNU noise makes the output of an imaging sensor noisy. In order to remove the effects of the noise due to the imperfections, the output model Y has to be denoised. In order to achieve this, a denoising filter (F) will be used. The resulting residual noise (W) after denoising is given in [47] as,

$$W = Y - F(Y) \tag{2.3}$$

$$= \mathrm{IK} + \tau\mathrm{D} + \mathrm{I} - \mathrm{F(Y)}\ \mathrm{c} + \emptyset$$

$$= IK + \tau D + £ \tag{2.4}$$

where £ is the modeling noise combined with the residual noise and it is given as,

$$£ = I - F(Y)\ c + \emptyset \tag{2.5}$$

Textured regions and near edges are usually characterized with large W. Several denoising filters including both non-adaptive and adaptive filters have been used in denoising a noisy image. The wavelet-based denoising filter proposed by Lukas et al. [5] has been used and reported to be a very good denoising filter for SPN extraction in [9, 10, 48, 49].

2.3. 2 Wavelet-Based Denoising Filter

Both non-adaptive and adaptive filters have been used for denoising images. Denoising filters such as Argenti's filter and Mihcak's filter are discussed in Amerini et al. [6]. The Argenti's filter models the signal-dependent noise, unlike the Mihcak's filter which models the noise by assuming it to have a zero mean and a stationary white Gaussian noise with an unknown variance. The parameters used in the proposed filter in this work are the variation of zero mean noise when it is yet to be correlated and variation of the noise associated with the electronic device. The Mihcak's filter, on the other hand, is a spatially adaptive statistical modeling of wavelet coefficients. The noise variance was calculated using MAP of the noisy image. Out of several denoising filters that have been used in literature, the denoising filter proposed by Lukas et al. [5] have been widely used and rated efficient in [9] and [49]. The aim of this section is to now summarize the basic steps involved in its implementation. This can be given by the four steps as stated below;

i. Wavelet decomposition of noisy Images: Noisy image was initially processed by adding Gaussian noise $N(0,\sigma^2)$ to the image. The vertical $v(i,j)$, horizontal $h(i,j)$ and diagonal components $d(i,j)$ were obtained using fourth-level wavelet decomposition with Daubechies of order 8.

ii. MAP of local variance: The local variance of the original image before the Gaussian noise was added was estimated for each component in each level using MAP for a neighborhood of 4 sizes. The least variance was used in the experiment. The estimated local variance using MAP is given in [5] as,

$$\sigma^2(i,j) = max\left(0, \quad \frac{1}{R^2}\sum_{(i,j)\in N} h^2(i,j) - {\sigma_0}^2\right), \qquad (i,j) \in R \tag{2.6}$$

where R is the decomposition level, σ_0 is the initial variance.

iii. Using Weiner Filter: The denoised image was obtained using Weiner filter for all the wavelets components and was repeated for all levels and channels (for true color images). The Weiner filter can be expressed for the horizontal component as,

$$h_{den}(i,j) = h(i,j)\frac{\sigma^2(i,j)}{\sigma^2(i,j) + {\sigma_0}^2} \tag{2.7}$$

iv. Applying inverse transform: This was used to reconstruct the denoised image.

For Section 2.3.3 to Section 2.3.8, we shall discuss the strengths and weaknesses of some of the state of the art methods which have been used for source camera identification.

2.3.3 Basic SPN

Basic SPN simply means the fundamental method upon which other methods of SPN extraction are based. This was based on the work done by Lukas et al. [5]. The first step is to extract the PRNU from photos captured from cameras. In order to extract a robust reference fingerprint for a particular camera, It must be estimated from the SPN of several images (30 and above). The SPN extraction for a single image has been explained in Section 2.3.1. Given $c = 1, 2, \ldots \ldots C$, $\mathrm{n} = 1{,}2 \ldots .\mathrm{N}$ and $\mathrm{W_n}$ is the PRNU of an image, the fingerprint of a camera can then be expressed as;

$$\widehat{\mathrm{K}}_{\mathrm{c}} = \frac{\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{W_n}}{\mathrm{N}}, \tag{2.8}$$

where C is the total number of cameras to be identified and N is the total number of images or photos from the single camera. The $\widehat{K_c}$ was based on finding the average PRNU over the number of images captured from camera c. During testing, images will be captured from each camera and their PRNU will be extracted. In order to verify whether a particular image is from a camera c, correlation is used as the decision parameter. The correlation parameter can be defined as;

$$p(i,j) = \frac{(n_i - \overline{n_\imath})(n_j - \bar{n}_j)}{\|n_i - \overline{n_\imath}\| . \ \|n_j - \bar{n}_j\|}, \tag{2.9}$$

where n_i is the PRNU of the camera fingerprint, $\mathrm{n_j}$ is the PRNU of the testing image and $\overline{n_\imath}$ and $\bar{n}_j$ are the average values of n_i and $\bar{n}_j$ respectively. The camera with the maximum correlation is taken to be the camera of the testing image. The proposed method by Lukas et al. [5] requires that cropping of images be carried out so as to achieve reduced computational complexity. This usually causes desynchronization and also affects the quality of the camera fingerprint which reduces the identification accuracies of the cameras. In order to increase the identification accuracies of cameras, several approaches have been adopted to enhance the camera's fingerprint. Chen and Fridrich [8] proposed another approach to estimate the camera fingerprint and this has been explained in Section 2.3.3.

2.3.4 Maximum Likelihood SPN

MLE was proposed by Chen and Fridrich [8]. In this work, instead of estimating the average of the PRNU over several images as the camera fingerprint, MLE was used as the camera fingerprint. Assuming the dark current to be insignificant, the noise residue in eqn. (2.4) can be further expressed as,

$$W = IK + £ \tag{2.10}$$

The log-likelihood of the noise term in eqn. (2.10) can be expressed as,

$$L(K) = -\frac{N}{2}\sum_{n=1}^{N} log\ (2\pi\sigma^2 \Big/ I_n^2) - \sum_{n=1}^{N} \frac{W_n/I_n - K}{2\pi\sigma^2/{I_n}^2} \tag{2.11}$$

The maximum likelihood estimate of K after the taking the partial derivatives of L(K), can be expressed as;

$$\widehat{K} = \frac{\sum_{n=1}^{N} I_n W_n}{\sum_{n=1}^{N} I_n^2} \tag{2.12}$$

The significance of eqn. (2.12) is that it gives a weighted sum of IK and this helps create a strong presence of the PRNU in the camera fingerprint. The quality of the estimated fingerprint (q) can be measured by estimating the correlation $(corr)$ between K and $\widehat{K}$. q can be defined by using the expression,

$$q = corr\ (K, \widehat{K}) \tag{2.13}$$

The PRNU are usually contaminated with artifacts due to in-camera processing techniques such as compression, quantization, and color interpolation. There are artifacts embedded in the PRNU due to the design of the imaging sensor, on sensor signal transfer. These artifacts are camera dependent, unlike PRNU which could still have a similar estimate for cameras of the same model. Most of these artifacts are caused by the demosaicing algorithms used in the cameras during color filter interpolation. Since these artifacts are periodic in nature, they are eliminated by the zero-mean operation. To zero-mean refers to finding the mean of each row or column and then subtract from each element in the rows and columns respectively [47].

Despite the improvement over the Basic SPN using MLE SPN, the limitation mostly associated with this method of SPN extraction is high contamination from scene details. The scene content of an image has its high-frequency components whose values are far greater than the frequency component of a clean SPN and therefore its presence highly contaminates the expected SPN. In order to combat the issue of scene content, Li [9] developed six models for enhancing the camera SPN. The next discussion focusses on the theoretical background upon which the proposed enhancing models are based, the summary of the author's achievements and current limitations of the models.

2.3.5 SPN using Li' Enhancing Models

For the SCI, 30-50 flat images are captured for each camera and the SPN noise for each camera is estimated as the average of SPNs. Natural images which may include high texture complexity from these cameras will be randomly captured and their SPN are extracted. The decision on a camera test image will be grossly affected if the extracted SPN of the test image consists of strong details of its scene content. Contamination by scene content reduces the camera identification accuracy. Secondly, the extraction of SPN will be computationally intensive if a huge number of images are used. Therefore, in order to reduce the computational complexity, small parts of the images are cropped to the same size and used in the extraction process. Cropping causes high variation in SPNs and also reduces the number of SPNs components compared to when the original size of the images are directly used in the extraction process. This also reduces the camera identification accuracy. Therefore, the quality of the extracted SPN determines whether high camera identification accuracy will be obtained. The work done by Chen et al. [50], attempted to enhance the SPN by attenuating artifacts caused by color interpolation, compression, row-wise and column wise operation. The work failed to deal with the limitation caused by the strong presence of scene content. SPN enhancing models are proposed by Li [9] to address the interference from the scene details.

The proposed models are based on the background knowledge that the magnitude of the scene details is higher than the SPN. The hypothesis for the models in [9] is that ''the stronger a signal component in n (the PRNU fingerprint), the more likely that it is associated with strong scene details, and thus the less trustworthy the component should be''. Weighting factors are therefore used to assign less importance to the scene components in the wavelet domain. The proposed six models are given in eqn. (2.14) to eqn. (2.19) respectively. The performance of each model is determined by the value of the threshold parameter (α) and $n(i,j)$ is the value of each signal component in row (i) and column (j) of n.

The model 6 given by eqn. (2.19) is usually avoided since it gives less significance to the weaker components and associates more importance to the scene details. Models 1 to 5 can be divided into three types. The types, corresponding models, threshold contribution to unenhanced camera's fingerprint in [9] are summarized in Table 2.1. It was reported by Li [9] experiment that models 1-5 all have good results but Model 5 has the highest peak rate of 1040 out of 1200 at a threshold value of 7. Phase sensor pattern noise has also been adopted in combating the contamination by scene details and also introduced peak to energy correlation as the decision parameter. This is explained in detail in Section 2.3.5.

$$\textbf{Model 1}:\ n(i,j) = \begin{bmatrix} \frac{n(i,j)}{\propto}, & \text{if } 0 \leq n(i,j) \leq \propto \\ e^{-0.5\frac{(n(i,j)-\propto)^2}{\propto^2}}, & if\ n(i,j) > \propto \\ \frac{n(i,j)}{\propto}, & \text{if } -\propto \leq n(i,j) < 0 \\ -e^{-0.5\frac{(n(i,j)-\propto)^2}{\propto^2}}, & \text{if } n(i,j) < -\propto \end{bmatrix} \tag{2.14}$$

$$\textbf{Model 2}: n(i,j) = \begin{bmatrix} \frac{n(i,j)}{\propto}, & \text{if } 0 \leq n(i,j) \leq \propto \\ e^{-\propto-n(i,j)} & \text{if } n(i,j) > \propto \\ \frac{n(i,j)}{\propto}, & \text{if } -\propto \leq n(i,j) < 0 \\ e^{-\propto-n(i,j)}, & \text{if } n(i,j) < -\propto \end{bmatrix} \tag{2.15}$$

$$\textbf{Model 3}: n(i,j) = \begin{bmatrix} 1 - e^{-n(i,j)}, & \text{if } 0 \leq n(i,j) \leq \propto \\ (1 - e^{-a})(e^{-\propto-n(i,j)}), & \text{if } n(i,j) > \propto \\ -1 - e^{-n(i,j)}, & \text{if } -\propto \leq n(i,j) < 0 \\ (-1 - e^{-a})(e^{-\propto+n(i,j)}), & \text{if } n(i,j) < -\propto \end{bmatrix} \tag{2.16}$$

$$\textbf{Model 4}: n(i,j) = \begin{bmatrix} 1 - \frac{n(i,j)}{\propto}, & \text{if } 0 \le n(i,j) \le \propto \\ -1 - \frac{n(i,j)}{\propto}, & \text{if } -\propto \le n(i,j) < 0 \\ 0, & otherwise \end{bmatrix} \quad (2.17)$$

$$\textbf{Model 5}: n(i,j) = \begin{bmatrix} e^{-0.5\frac{n^2(i,J)}{\propto^2}}, & \text{if } 0 \le n(i,j) \\ -e^{-0.5\frac{n^2(i,J)}{\propto^2}}, & otherwise \end{bmatrix} \quad (2.18)$$

$$\textbf{Model 6}: n(i,j) = \begin{bmatrix} 1 - e^{-n(i,j)}, & \text{if } 0 \le n(i,j) \le \propto \\ (1 - e^{-a})(e^{-\propto + n(i,j)}), & \text{if } n(i,j) > \propto \\ -1 + e^{-n(i,j)}, & \text{if } -\propto \le n(i,j) < 0 \\ (-1 - e^{-a})(e^{-\propto + n(i,j)}), & \text{if } n(i,j) < -\propto \end{bmatrix} \quad (2.19)$$

Table 2.1. Model types and threshold contribution to un-enhanced SPN

Types of Transformation	**Model(s)**	**Threshold's Contribution**
Linear	1 and 2	It gives the exact weights to n in the within -α to α. It is the most conservative.
Non-Linear	3	It gives greater significance to SPN components on the lower ends and less significance when closer to ±α. It is a moderate operation.
Inversely Proportional	4 and 5	It interchanges the order of the SPN magnitudes. Weaker components are given greater significance and it is considered to have a significant effect on the existing magnitudes.

2.3.6 Phase SPN

The Phase SPN pattern noise proposed in [10], also addresses the problem or effect of scene details on the PRNU. The approach explores the receiver operating curve characteristics to suppress the effect of the contamination by the scene content.

The phase component of the noise residue W_j is obtained by taking the phase component of the DFT of the SPN. Given, $W^{j'} = DFT(W_j)$, the phase component of the PRNU can be expressed as,

$$W_{\emptyset j} = \frac{W_j}{|W_j|}, \tag{2.20}$$

where, W_j is the SPN of camera j. The overall SPN of a camera is then estimated by calculating the average of all the phase components of SPN of each image from a camera and then find the inverse DFT. This can be expressed as;

$$y = real(IDFT(\frac{\sum_{j=1}^{N} W_{\emptyset j}}{N}), \tag{2.21}$$

where y is the camera reference SPN and N is the total number of images per camera.

2.3.7 Weighted Averaging Based SPN

The work in [51] proposed another approach to account for the variation in the SPN estimate using weighting averaging. The motivation behind their method is that the random noise in SPN of the camera does not has the same variance across all images captured by the same camera device. Out of the several other reasons, camera settings, focal length and shutter speed contribute to the variation in random noise of PRNU. The goal of their proposed weighting averaging is to minimize the estimation error due to this variation. The sum of the signal in a random noise, $(S(j))$ is expressed in [51] as,

$$S(j) = \sum_{i=1}^{N} w_i\, s_i\,(j)\ , \tag{2.22}$$

where w_i is the optimal weight for ith observation, $s_i(j)$ is the sum of $S(j)$ and the variance of each observation (σ_i) and zero mean of the observation, $s_i(j) = S(\text{j}) + r_i(\text{j})$, $i = 1, \dots N$, $j = 1, \dots L$, r_i(j) is the random noise. Note that the s_i (j) is column vector of the given image, i. The optimal weight for ith observation is expressed in [51] as,

$$n_\sigma(j) = s_i\,(j) - \overline{s_i}\,(j) \tag{2.23}$$

The noise variance (n_σ) was proposed as,

$$w_i\ = \frac{1}{\sigma_i^2}\left(\frac{1}{\left(\sum_{i=1}^{N}\frac{1}{\sigma_i^2}\right)}\right) \tag{2.24}$$

In order to obtain the weight, we have to estimate (σ_i) and this is expressed as,

$$\sigma_i\ = \sqrt{\frac{\sum_{j=1}^{L}\left(n_\sigma(j) - \overline{n_\sigma}(j)\right)^2}{L}} \tag{2.25}$$

The method in [51] proposed the weighting averaging approach with two the basic SPN and also MLE SPN and both are called weighted Basic SPN and weighted MLE SPN respectively. Given that the PRNU of a given observation is r_i , the weighted Basic SPN can be expressed as,

$$weighted\ Basic\ SPN = \frac{\sum_{i=1}^{N} w_i\, r_i}{N} \tag{2.26}$$

Also, the weighted MLE SPN can be expressed as,

$$weighted\ MLE\ SPN\ = \frac{N(\sum_{i=1}^{N} w_i\, r_i I_i)}{\sum_{i=1}^{N} {I_i}^2} \tag{2.27}$$

where, I_i is the intensity of each given image of a camera.

2.3.8 Summary of Other State of the Arts Methods for Source Camera Identification

Apart from the four described commonly compared state of the art methods, there are other approaches that have been proposed in literature. The work in [8] and [52] proposed the removal of sharing components in the PRNU estimate. The motivation of their works was based on the fact that there are artifacts of pipeline processing of cameras occurring in the estimated PRNU. In order to suppress this artefact, they proposed the use of zero mean operation coupled with the use of Wiener filtering of the PRNU estimate in Fourier domain. The aim of transforming features to Fourier domain is to suppress the magnitude of the artifacts and hence strengthening the presence of the PRNU. On the premise that the PRNU fingerprint is a Gaussian noise, the work of Li and Lin [11] proposed the use of spectrum equalization algorithm. In their work, they proposed that SPN is not likely to be periodic but more likely to have flat spectrum. Therefore, unlike the work in [8] and [52] that attempted to suppress the magnitude of the artifacts in the Fourier domain, they used an iterative means to detect unwanted peaks in Fourier domain. After the detection of these peaks, they adopted smoothing of the spectrum of camera images by obtaining the mean of local spectrum components. This was done to remove periodic artifacts and thereby, increasing the quality of the SPN which will further increase the identification accuracies of cameras. Furthermore, the work in [12] proposed the use of improved locally adaptive filtering instead of the widely used wavelet denoising filtering. Also, they estimated the PRNU using true color images with the use of weighted averaging proposed in [51] for the PRNU estimation. The work in [12] reported superior performance than the use of wavelet denoising filter with weighted averaging and also with other compared state of the art methods. Instead of wavelet-based transform, the work in [53] adopts dual tree complex wavelet transform while the work in [54] adopts coupling and adaptive filter. Furthermore, the work in [55] proposed the enhancement of the PRNU pattern alignment while the work in [56] proposed a low dimensional PRNU features for effective SCI.

Moreso, some deep learning methods based on convolutional neural networks have been proposed in literatures [19, 28, 29, 38, 57-59] for source camera identification and some have been reviewed under motivation for this work in Section 1.2.

2.3.9 Comparative Analysis of Some Existing SPN Extraction Method on the Effect of Images Size on Identification Accuracy

The motivation behind this section is to carry out an experiment using some of the states of the art method to establish the effect of PRNU of small image sizes on camera identification accuracy. The motivation behind this is to establish our objective of proposing deep learning approaches for small image since PRNU fingerprint becomes weak for small image sizes. The reference SPN for each camera is prepared using 50 images per camera. 50 flat and natural images were used per camera for both training and testing phases respectively. The images were cropped at different resolutions including 128×128, 256×256 and 512×512. The reference SPN is the compared with 50 test images per camera. There are different decision parameters usually used for the matching between the reference PRNU of cameras and the PRNU of their testing images. These include normalised correlatiion, peak to energy correlation (PCE) and circular cross-correlation norm (CCN). CCN is the improved version of PCE. In PCE, only positive values of correlation between the PRNU fingerprints of cameras and the PRNU of cameras of testing images were considered. However, CCN also takes into account, the negative values of correlation [10]. Normalized correlation uses varying thresholds while PCE and CCN use constant threshold [60]. PCE and CCN further suppresses the problem of periodicity in SPN than when normalised correlation is used. Moreso, in [61], channel-wise correlation is used for source camera identification with PRNU-based techniques. This means that the correlation coefficient is calculated for each image channel (Red, green and blue). This is found to achieve better result, and this is associated with the fact that the green component consists of a significant amount of the SPN signal. In this experiment, PCE is adopted as the matching parameter and can be defined as the ratio of the squared correlation divided by sample variance of the circular cross-correlations. The PCE can be expressed as,

$$\mathrm{PCE}\left(\mathrm{I}, \widehat{\mathrm{K}}\right) = \frac{\max\,(\mathrm{p})^2}{\frac{1}{\mathrm{T}-\varepsilon}\sum_{\mathrm{t}=1\ldots.\mathrm{T}\notin\varepsilon}\mathrm{p}[\mathrm{t}]^2} \tag{2.28}$$

where, T is the sum of the pixels' intensity in the testing image, p is the cross-correlation between the reference and the test image, ε is a small number of correlation entries around the maximum number of p.

2.3.10 Experimental Settings

Table 2.2. Cameras Used in Experiment, Sensor's Type, Resolution and Image Format

Cameras	Sensor Size (mm)	Native Resolutions	Image Format
Canon_Ixus70_0	1/2.5”	2304×3072	JPEG
Canon_Ixus70_1	1/2.5”	2304×3072	JPEG
Sony_DSC-H50_0	1/2.3”	2592×3456	JPEG
Sony_DSC-H50_1	1/2.3”	2592×3456	JPEG
Nikon_CoolPixS7	1/1.73”	3264×4352	JPEG
Samsung_L74wide	1/2.5”	2304×3072	JPEG

Table 2.3. Experimental Parameters and Descriptions

Parameters	**Descriptions**
Format	JPEG
Number of reference images per camera	50
Number of test images per camera	50
Number of cameras	6
Cropped images sizes	128 by 128, 256 by 256 and 512 by 512.
Nature of reference images	Flat Images From Dresden Database
Nature of test images	Natural images (including dark images)
Peak Threshold value for Model 4 and 5	18

Table 2.4. Identification accuracies of the four SPN methods (%)

Image Resolutions	Methods	Canon_Ixus70_0	Canon_Ixus70_1	Sony_DSC-H50_0	Sony_DSC-H50_1	Nikon_CoolPixS710_0	Samsung_L74wide_0	Average accuracies
512×512	Basic SPN	100	100	100	100	100	100	100
	MLE SPN	100	100	100	100	100	100	100
	Phase SPN	98	98	100	100	100	100	99.3
	Model 4	100	98	100	96	94	98	97.7
	Model 5	100	100	100	98	100	100	99.6
256×256	Basic SPN	94	94	100	100	92	100	96.7
	MLE SPN	94	96	100	100	98	100	98.0
	Phase SPN	94	96	100	100	90	96	96.0
	Model 4	88	90	98	94	84	90	90.7
	Model 5	96	94	100	96	96	100	97.0
128×128	Basic SPN	82	86	96	92	74	76	69
	MLE SPN	84	86	100	92	76	76	85.7
	Phase SPN	88	82	98	92	68	80	84.7
	Model 4	78	70	94	80	62	62	74.3
	Model 5	84	84	98	86	76	70	83.0

2.3.11 Results and Discussion

The results for the four described methods are shown in Table 1. The models 4 and 5 are achieved when $\alpha = 18$. From the experiment, it can be deduced that MLE and Model 5 of Li's models performed better. Also, the identification accuracy decreases up to 62% at 128 resolutions. The work done so far shows the camera identification accuracy using PRNU decreases with increase in the size of the image. Also, the identification accuracy can be very low for images with high texture complexity.

Chapter 3 :

Review of Deep Learning Algorithms and Techniques

3.1. Historical Background of Artificial Neural Network

Artificial intelligence is the science that involves making machines to think like humans. In 1943, McCulloch and his co-author Walter Pitts proposed the brain neurons to have similar behavior to a tuning machine. Based on the experiment carried out by them, they came to the discovery that the brain neurons are more than a two-state network. An artificial neural network (ANN) was developed. This was further developed by Donald Hebb in 1948 by bringing an alteration to the already proposed ANN so as to explain how the ANN learns. This he was able to achieve by associating an independent weight to each input of the network. In 1962, another learning algorithm called perception was introduced by Rosenblatt in 1962. Perceptron is used for the classification of a given set or situation into two groups. This is done by viewing them on a hyperplane using some functions once they could be made to be linearly separable. This means perceptron has a limitation in classifying sets of groups when they are not linearly separable. Also, it could not be used to represent inputs that are dependent on XOR and XNOR for their generation [62].

In the meeting of researchers in Dartmouth in MIT, John McCarthy later came up with the term artificial intelligence. It was in this meeting, the principle of the convergence theorem was further substantiated as a learning algorithm which could be used to complement the weakness in the use of perception [63]. By the year 2000, more researchers found out that human machines could be extended beyond human intelligence to more real-time world problems. Minsky and Papert [64], it was reported that the use of neural networks gradually becoming limited due to the fact that they becoming inefficient to learn deep features and hence results to low discriminating power against testing inputs. ANN is, therefore, referred to as a shallow architecture. The basic principle of ANN will be more explained in Section 3.2.1.

3.2.1 Artificial Neural Network

In order to build an ANN, the number of neurons, the associated weights, biases, activation function and also the learning algorithm for training the network are essential things to be considered. The neuron is a core element in the processing of information. In order to explain the working principle of ANN, we will consider the simplest ANN shown in Figure 3.1[34]. It consists of only one neuron and with three inputs; x_1, x_2, x_3 and the intercept term, b. The associated weights of the input values are denoted as w_1, w_2, w_3 and $h_{w,b}(x)$ is the output of the neuron. The output $h_{w,b}(x)$ depends on the kind of activation function used.

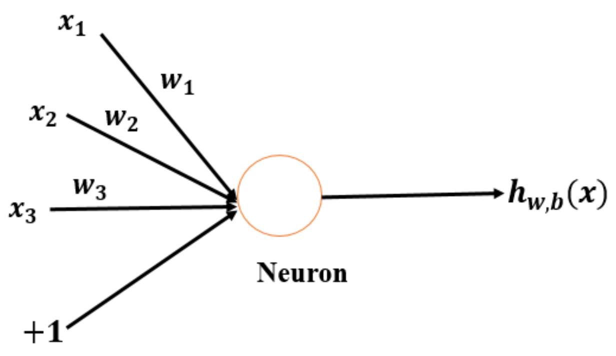


Figure 3.1. Modified Diagram of a Single Neuron.

The output of the network in Figure 3.1, can be expressed as,

$$h_{w,b}(x) = f(\sum_{i=1}^{n} w_i x_i + b) \tag{3.1}$$

where $f(.)$ is the activation function.

The neural network architecture is formed by the combination of several simple neurons. The output of a neuron is usually fed as the input to the corresponding neuron. We use the Figure 3.2 to explain the formation of the neural network. Figure 3.2, consists of a neural network with three layers. The layers include the input layer (Layer 1), the hidden layer (Layer 2) and the output layer (Layer 3). The layers are denoted by L_1, L_2 , L_3 , respectively.

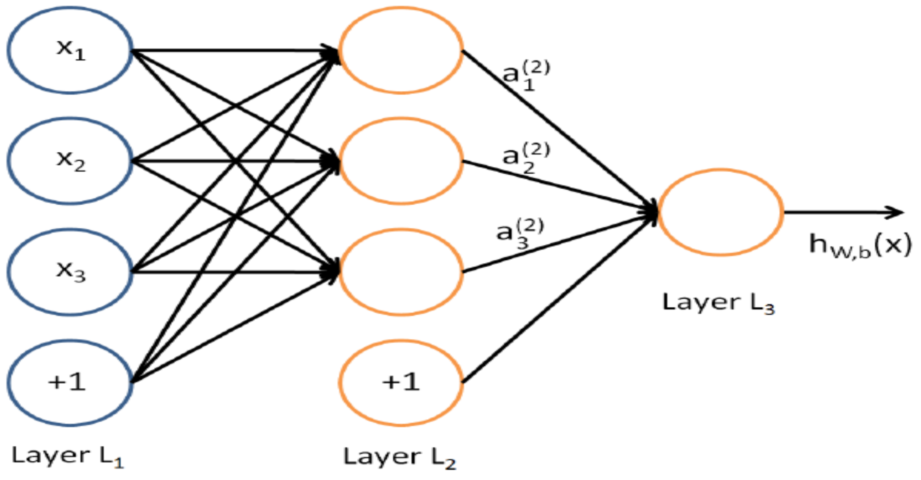


Figure 3.2. Three Layered Neural Network Architecture [34].

In the first layer, three are inputs and a bias term. This layer is referred to as the input layer. It should be noted, in most network architecture, we use the bias term to be +1 so as to leave nth actual inputs to the neurons. Each of the inputs is now connected to the neurons of the next layer excluding the bias of the layers. This second layer is called hidden layer. It is called hidden layer because its outputs are hidden both to the input and the output layers. The layer 3 with a single output for the case of this example is called the output layer. It should be noted that the output of a neural network could be more than one. The numbers of neurons in a layer excluding the intercept term are called the units of the layers. All the units in a layer are connected to the next layer through weights. Each layer has its own collection of weight and is represented in form of a matrix. Given that l denotes a particular layer, the weights associated to the later can be denoted as W^l. The two parameters of the network are the weights and the biases and are usually represented as (W, b). For the ANN under consideration, (W, b) denotes(W^1, b^1, W^2, b^2) for a three-layered ANN. The output of a layer is hypothesized using an activation function and they are represented as a^l. These new outputs from each layer are referred to as the learned features. These learned features serve as inputs to the next layer [34]. Minsky and Papert [64], as stated initially, reported that the use of neural networks gradually becoming limited due to the fact that they becoming inefficient to learn deep features and hence results to low discriminating power against testing inputs. This necessitated the learning of deeper architectures.

3.2.2 Activation Functions

There are different forms of activation functions that help to map the input to an output. Mostly used activation functions in most learning algorithms are n on-linear functions [34]. The most commonly used in deep learning algorithms are a sigmoid or logistic function, Tanh function and the rectilinear unit function (ReLU). The sigmoid function can be expressed as,

$$f(z) = \text{Sigmoid function} = \frac{1}{1 + \exp(-z)} \tag{3.2}$$

The hyperbolic or Tanh function can be expressed as;

$$\mathrm{f(z) = Tanh\ function} = \frac{\mathrm{e^{z} - e^{-z}}}{\mathrm{e^{z} + e^{-z}}} \tag{3.3}$$

Lastly, the ReLU is given by the expression;

$$\mathrm{f(z) = ReLu = max\,(0, z)} \tag{3.4}$$

where z is the weighted input or net input. It should be noted that sigmoid function is usually used when input values ranges between 0 and 1 while Tanh function is usually used for input values with the ranges of -1 to 1. Another activation function very close to the Tanh function is the Sofsign function ($f(z) = (1/1 + |x|)$ [65]. Unlike Tanh, it uses polynomial. It also produces smoother asymptotes compared to Tanh function. It works with input values that ranges around 0 and also -1 and 1. Therefore, it has both linearity and non-linearity properties. In Krizhevsky et al. [66] , the ReLU, was reported not to be the global optimal result required but several combination of it may yield good result and satisfactory efficiency. In [67], it was also reported to yield better performance and faster convergence speed when trained by backpropagation and also as it has the tendency of yielding higher classification accuracy. ReLU activation is mostly used when the neural network is of a very large inputs. Its ability to generate a sparse representation makes it achieves higher efficiency [68]. It should be noted that the logarithm of the sigmoid function in eqn. (3.2) is called log sigmoid and it is usually used instead of the normal sigmoid function so as to make its computation faster. Other forms of non-linear activations used for deep learning modules include the Maxout, Softplus and, Soft sign functions. Maxout function outputs the maximum unit at each update in the hidden layer. According to [69], Maxout and Tanh functions are most applicable to the nature of data that satisfy the negative saturation property. The Softplus function has smoothness property and this enables it to be suitable for sparsity regularizations in deep learning modules such as stacked autoencoders [70]. Its smoothness property is due to its having a positive bias gradients. Generally, ReLU, Softplus, and Sigmoid functions produce a robust and invariant representation of features when as activation functions for networks with sparsity regularizations.

3.2. Introduction to Deep Learning and its Challenges

The need to extract robust features has prompted the need to explore more deep architectures rather than shallow architectures that have been earlier adopted in many applications. Any neural network with just a single hidden layer can be referred to as shallow architecture. Other examples of shallow architectures include SVMs, kernel regression, and multilayer perceptron. The shallow architectures are limited in that they have limited hidden layers and hence, has less efficiency [71]. It has been reported in that deep architectures are always more efficient than shallow architectures due to their ability to represent the input data with more robust features [72]. Digital signal processing including speech, image, and video signals requires deep architectures for them to be able to extract robust or rich features capable of having very high discriminative power for classification purpose. Several non-linear processes are involved in deep architectures and the output of a layer is fed as input to the next layer for deeper features extraction. This makes it possible to learn deep features based on the several abstraction levels and the extracted features are robust for classification problems. Deep neural network (DNN) can be defined as a network consisting of many layers through which feature hierarchies are learned. It will also be necessary, to know the background or the origin of deep classifiers [26].

The first developed deep network was used for the classification of handwritten digits. This was achieved using ordered or ranked multi-layered network. Despite the use of deep architectures, the obtained accuracy was low compared to using ANN with just one or two hidden layers. The limitation in the use of the developed deep network was due to difficulty in the training of features across different abstraction levels of the network. Though backpropagation with gradient descent had been developed since for training NN [73] but there was still higher classification error and this was attributed to unavailability of sufficient data for pre-training of network and also deficient algorithms [74]. Due to this limitation, the SVM was particularly used for training of NNs. The major challenge with the use of SVM is that it is only efficient for data that are linearly separable on a hyperplane. It cannot adequately learn NN with complex architectures and majorly applicable for linear data.

Despite these limitations, several types of research were still in progress regarding improving the performance of deep networks [75]. In 1992, Schmiddhuber [76] developed deep belief network that could learn a compressed representation of input data and it uses backpropagation with gradient descent as the optimization algorithms. However, the approach has the limitation of vanishing gradient. The introduction of the unsupervised learning algorithm in 2006 by Hinton kick-started the first breakthrough in overcoming some of the previously mentioned limitations [71, 77]. By unsupervised learning, we mean using learning algorithms that the pre-training did not involve the pre-knowledge of the targets of the classes involved in the training. Specifically, in their work, they made use of deep generative models [77]. The basic concept behind this approach is called the layer-wise pre-training. This has to do with training an input signal through several hidden layers without the knowledge of class levels. The idea is to learn robust features at each layer of the network. These features are called deep features. The output features at each hidden layer are extracted and fed in as the input to the next hidden layer. The complexity of a deep neural network is linearly proportional to the number of the hidden layers that are used in the developed deep architecture. The optimisation of weights at each hidden layer of the network is dependent on the learning algorithm adopted. Finally, their unsupervised pre-training algorithm will then be fine-tuned by connecting multilayer perceptron classifier with backpropagation at the last layer to predict the output [77].

However, there are some limitations with their approach due to weights optimization of the network. Firstly, the complexity of the weight optimisation by teasing especially, the top two layers results in overfitting during the training process. Secondly, the features in the lower abstraction levels may not be useful and connecting them with a supervised network can be tedious and difficult in tuning to achieve higher discriminating power [77]. The classification accuracy of a deep architecture can be improved by careful scaling of the input features and also the kind of activation function used [77]. In order to resolve the problem associated with weights optimisation in the DNN, In [78], Hessian-free second order methods have been proposed to resolve the vanishing of the gradient when pre-training is done layer to layer. For recurrent networks and the restricted Boltzmann machine, unsupervised pre-training has also been widely used in resolving the problem of weight optimisation [79, 80]. It should also be noted that good result can also be achieved when features are correctly ranked [81]. This has a way of improving the gradient obtained since there are millions of parameters used for training. Most of the methods described above used a constant learning rate during the learning process but in Cho et al [82], controlled learning rates were used for training an RBM. It has also be shown in the work done in [83], that the performance of DNN can be significantly improved by the correctly choosing the initial parameters in the first hidden layer. It has also be shown in several kinds of literature [65, 84] that the number of units used in each hidden layer greatly determines the robustness of the features and eventually the classification error. As have mentioned earlier before, DNN is computationally expensive. In order to reduce the number of features used in processing, it should be noted that not all features of an image are used for classification. There are some certain features that are in some spatial regions of images are good for generalisation. The features at these regions can be aggregated statistically. This process of aggregating features is called pooling. The common pooling techniques are the mean and max-pooling. This has also be shown in [79] to have a significant contribution to achieving high discriminating power during testing. Also, it has been reported in [85] that stochastic gradient on mini batches of several training examples is the best optimisation technique. Due to computational complexity, stochastic gradient using mini batches on large dataset are implemented online. Furthermore, when large dataset with labels is available, it has been reported that supervised training can replace by supervised learning with the same or little less performance compared to the unsupervised pre-training. An example of the work done using this approach is the ImageNet task. Due to a large dataset, it was accomplished by a combination of several traditional computers. In their [85], more than a 100 million training examples are used and this was made evident through the use of an efficient graphics processing unit (GPU) [66]. Another variant of autoencoder is denoising autoencoder. AEs adds noise to the inputs to the layers and this has a way of learning robust features by expanding the inputs. This has been reported to yield a very significant result when compared with just an encoder. The ability of a DNN to automatically learn features has made it to be widely adopted for several applications including hand digit recognition, face recognition [20], image classification [21], fingerprint liveless detection [22], automatic malware signature generation and classification [23], modulation format identification in coherent receivers [24] and classification of hyperspectral images [25]. As far as we know, it has not yet been widely adopted for camera identification. Though deep learning methods such as convolutional neural network have been used for camera model identification deep learning for source camera identification is still at a preliminary stage. Some of the objective functions and optimisation techniques used in deep learning algorithms are discussed in Section (3.3) and Section (3.4) respectively.

3.3 Objective Functions for Deep Learning

The objective functions are also called the loss functions, optimization score functions or cost functions. The objective functions are usually used as a measure of mismatch between an input data and the reconstructed data when given into a neural network. It could also be used to estimate the error between training data and the testing data. The goal is so that the minimum error is incurred during training. The mostly used conventional cost functions include the sum of the squared errors (SSE), cross-entropy (CE) and the exponential cost functions (EXP) [86, 87]. Classification error is also a measure of mismatch between a training data and the target. The classification error is defined as the ratio of the number of misclassified target samples that belong to a class to the total number of samples of that class. The SSE is defined as the sum of the squares of the difference between a training data and the reconstructed data or the target data. Cross-entropy, on the other hand, uses adaptive sampling to estimate the error through stochastic optimisation [88]. It measures the average number of wrong predictions between the probability distributions of the training and the target or reconstructed input in the case of autoencoders. The EXP cost function is a derivative of error entropy. It incorporates the ideas of both cross-entropy and the square mean square error. The SSE, CE and EXP cost functions can be expressed as given in eqn. (3.5) to (3.7).

$$L_{SSE}(x, \hat{x}) = \sum_{n} (x_n - \hat{x}_n)^2 \tag{3.5}$$

$$L_{CE}(x, \hat{x}) = \sum_{n} (x_n \ln(\hat{x}_n) - (1 - x_n) \ln(1 - \hat{x}_n)) \tag{3.6}$$

$$L_{EXP}(x, \hat{x}) = \mu \sum_{n} \frac{1}{\mu} (x_n - \hat{x}_n)^2 \tag{3.7}$$

where, x_n and $\hat{x}_n$ are the nth input and output vectors respectively. μ is the extra parameter added to eqn. (3.5) to form eqn. (3.7). In [86], comparisons were made for SSE, CE and EXP using stacked autoencoders. Based on the experimental results in [86], it was reported that SSE is a good cost option for pre-training process. The work in [86] reported SSE to give the lowest error and also variances between the hidden layers of the stacked autoencoders. Also, the mean square error decreases progressively down the layer and this simply means the reconstruction error of the learned features decreases with increased hidden layers. For the case of stacked autoencoders, the CE and the EXP were reported to have less performance compared to SSE. In [86], further comparison was made between the SSE and CE. One of the advantages of average cross-entropy as stated in [89], the presence of the $\ln(.)$ function. This function takes into an account, the closeness of predictions unlike, the classification error that only considers wrong predictions. For this reason, CE is considered to be more detailed compared to SSE. As earlier stated, SSE is considered to be a good choice of loss function when used with stacked autoencoder. However, wrong predictions were given more emphasis. In summary, the classification error only considers the wrongly classified targets and therefore is a very crude approach for measuring generalisation error. However, SSE and CE are usually used as the loss functions during the training process while the classification error is preferable as the loss measure between the trained data and the testing data. This is because during, generalisation, the interest is to correctly predict the test targets. For neural networks where backpropagation is used for fine-tuning, there will be a need to calculate the gradients of the cost functions and SSE and the CE affect the calculation of the gradient. Therefore, for most applications that adopt NNs, the SSE and the cross-entropy are considered suitable to measure the loss measure during training while the classification error is suitable for estimating the network generalisation error. Also, the cross-entropy loss is usually used when softmax classifier is used for supervised fine-tuning because it enables the changes in weights of the network not to have the tendency of decreasing to zero. When softmax classifier is used with cross-entropy loss, it is usually called the categorical cross-entropy loss.

3. 4 Supervised Optimisation Algorithms for Deep Learning

Optimisation techniques are used for updating hyperparameters needed for training DNN. There are many optimisation algorithms that have been explored for training DNN. The optimisation methods that have been used include Newton Rapson's method, Broyden Fletcher Goldfarb Shannon algorithm, Conjugate Gradient and stochastic gradient descent (SGD) based learning [79, 80]. The updates of parameters by these methods are usually based on a continuous search for the global minimum that will yield the lowest cost of the objective function [90]. There are other parameters used for DNN modules but the parameters that are used for learning process prior to the DDN models are called the hyperparameters. Examples of hyperparameters include the learning rate, mini-batch size, the number of epochs used for training DNN and L2 weight regularization parameter [91]. Each of the optimisation algorithms has both strengths and limitations. Also, some are suitable under some conditions while others are not [91]. The stochastic gradient-based learning algorithms have been widely used [92, 93]. This is due to their simplicity and low computational cost especially for problems with large training samples. One of the limitations with the use of SGD is the difficulty in setting its hyperparameters so as to achieve global minimum when minimising the cost function. SGDs involve trial and error method and as a result, it makes SGDS to be computationally ineffective. In order to address this weakness, validation set of sample is usually used for modeling and parameter selection. Also, SGDs are difficult to be trained using GPU [90]. This is because GPU involves parallel computing and SGDs have internal computations without a specific order. Despite these limitations, SGDs are still effective and easy to implement for large training samples. In [71, 77], SGDs are considered inappropriate for training DBNs over several hidden layers. The limited Broyden Fletcher Goldfarb Shannon and conjugate gradient are considered to be better optimisation algorithms than the SGDs for layer-wise pre-training because convergence is easy to be attained and can also be trained on GPUs. Provided that there is the availability of large computer clusters, multicore GPU and central processing unit, these methods are considered to be computationally slow. This is because a single update requires computation over the entire training samples [94]. This weakness can be addressed using the mini-batch training procedure. This has to do with updating parameters using a fraction of the training examples. This approach is now considered too computationally fast for large training datasets. It should also be noted, that the appropriate optimisation technique to be used may be dependent on the type of DBN module or the size of the dataset. Limited Broyden Fletcher Goldfarb Shannon algorithm has been considered to be highly suitable for low dimensional images and usually adopted with problems using CNN. Conjugate gradient, on the other hand, is good for high dimensional images [90]. Despite the limitations of SGDs, they still have low classification error when combined with line search and large mini-batch training. Also, the experiment carried out on the MNIST dataset in [90] shows that SGDs require that the learning rate has to be low in order to capture useful features at the hidden activation for noisy inputs. As stated in [95], the learning rate is usually considered as the most significant learning parameter to achieve the lowest classification error rate. The backpropagation with gradient descent method was used in [75] for automatic voxel-wise brain parcellation based functional connectivity using SSAE.

However, the gradient descent is considered to be one of the most widely used algorithms in most of the deep learning modules. In this section, we will review some strengths and weaknesses of gradient descent optimisation algorithms. There are several gradient descent variants used in deep learning techniques. The goal of the optimisation algorithm is to minimise the objective or cost function ($J(\theta)$). θ represents the parameters of the cost function. The idea is to obtain the gradient of the cost function to be zero. This goal is practically not achievable but we can decrease the cost function close to zero, till we can obtain a local or global minimum. There are several variants of gradient descent optimisation algorithms such as the batch gradient descent, stochastic gradient descent, mini-batch gradient descent, momentum, Nesterov accelerated gradient, Adagrad, Adadelta, RMSprop and Adam. In the following sub-sections, the overview of the ideas and limitations of these optimisation methods [96] will be discussed.

3.4.1 Batch Gradient Descent

In batch gradient descent (BGD), a single update is carried out over the dataset. This can be expressed as given in eqn. (3.8). η is the learning rate and ∇_θ is the gradient of the cost function. The number of steps to be taken to reach a local or global minimum is determined by the η. One of the disadvantages of the BGD is very slow computation since only update over the entire dataset. This also makes it difficult for online training. Despite the computation demand, there is higher probability for the gradient to converge at a global minimum, unlike some other methods that could stick at a local minimum.

$$\theta = \theta - \eta . \nabla_\theta J(\theta) \tag{3.8}$$

3.4.2 Stochastic Gradient Descent

Unlike the BGD, In SGD, an update is carried out for each sample in the dataset. This helps to eliminate the problem of redundant computation demands of BGD. It makes it possible to perform online training since, for every single update, the gradient is computed. Given that n is the index of the training sample, $\mathrm{x(n)}$ and $\mathrm{y(n)}$ as the training data and targets respectively, SGD can be expressed as,

$$\theta = \theta - \eta . \nabla_\theta J(\theta, x(n), y(n)) \tag{3.9}$$

The advantage of the SGD is that, randomness compared to using just a single update as in BGD. It is faster to implement. Despite these advantages, the SGD has the tendency of missing the global minimum because of overshooting. In order to aid convergence of SGD, the learning rate has to be very small so as to increase the number of steps to be taken over the entire dataset. Shuffling of the training dataset for each update is an essential step to further enhance the optimization process.

3.4.3 Mini- Batch Stochastic Gradient Descent

In order to reduce the training time for huge training dataset, another approach is to divide the entire training dataset into groups called mini-batches. This approach makes the optimization faster compared to both the BGD and SGD. How faster an update is done is determined by the mini-batch size (N_B). The mini-batch stochastic gradient descent (MBSGD) can be expressed as,

$$\theta = \theta - \eta . \nabla_\theta J(\theta, x(n\!:n + N_B), y(n\!:n + N_B)) \tag{3.10}$$

The mini-batch size to use vary for different applications but for most deep learning applications, the mini-batch size usually set with the range of 50 to 256. Another advantage of using mini-batch gradient descent is that it makes the implementation of parallel computing easier to be carried out using the GPU. Though SGD can also be implemented on GPU but it is more demanding due to the need for synchronization after every sample update.

3.4.4 Momentum

Despite the benefits of using mini-batch gradient descent, obtaining convergence to global minimum is not guaranteed. There are still some challenges to combat so as to enhance the optimisation process. One of such challenges is how to optimally select the learning rate. The learning rate is a principal factor that determines the rate of convergence to either local or global minimum. Too small learning rate means that it will require a longer time to reach convergence and too large learning rate could lead to jumping or skipping the global minimum. Therefore, one of the gradient descent-based optimisation technique is the use of momentum. The word momentum literally means impetus for a given object. Therefore, the function of momentum is that it helps to speed up the SGD optimisation. It adds a fraction of a past update (γ) to the next update. The new estimation of parameters can be expressed as,

$$\theta = \gamma\theta - (1-\gamma).\nabla_{\theta}J(\theta, x(n), y(n)) \tag{3.11}$$

Momentum helps to increase the rate at which global minimum is reached and also helps to dampens oscillation.

3.4.5 Nesterov Accelerated Gradient

There is a need to control the rate at which the momentum aids the rate of convergence to the global minimum. This is to regulate the rate at which the updates speed up to convergence.

Nesterov accelerated gradient helps to give a fore-knowledge to the momentum term. Given a momentum term (γ), in order to calculate the rate of changes in the position of the parameters for the new update, a rough estimation of the new set of parameters are made using the previous parameters. The next parameters to be estimated can be expressed as,

$$\theta = \theta - v_n \tag{3.12}$$

where v_n can be expressed as,

$$v_n = \gamma_{v_{n-1}} + \eta. \nabla_\theta J(\theta - \gamma_{v_{n-1}}, x(n), y(n)) \tag{3.13}$$

The aim of the Nesterov accelerated gradient is to make necessary corrections where there is a big jump in the gradient of the cost function while using momentum. This helps to improve generalisation performance. It is reported in [96], that Nesterov accelerated gradient improved the performance of recurrent neural networks.

3.4.6 Adagrad

Adagrad means adaptive gradient descent. This simply means that the learning rate is not constant for all updates. When learning a dataset, not all the features of the dataset have greater contribution to guarantee better generalization are achieved during training instances. These features show up occasionally and therefore, they are referred to as infrequent features. There are also features that carry less information necessary for better performance and may show up more frequently during training instances. These features are called frequent features. In order to ensure more significance are given to the infrequent yet important features, the higher learning rate is attributed to them so as to increase the number of updates for those features. Adagrad is usually suited for datasets with sparse features because, in most applications, sparse features are more useful for improved performance. Therefore, the usefulness or the suitability of the Adagrad optimization algorithm is also dependent on the nature of the training data. In Adagrad, instead of using the same learning rate to learn parameters, the learning rate is changed at every given step size, l. Given that $i = 1,2, \dots l$, the newly learnt set of parameters can be expressed as,

$$\theta_{l+1,m} = \theta_{l,m} - \frac{\eta}{\sqrt{\nabla_{\theta} J(\theta_m)}} \tag{3.14}$$

The value of the learning rate usually used in literature is 0.01[96]. The learning rates for all the step sizes are automatically set. One of the disadvantages of using Adagrad is that the learning rate decreases to a point where important features could no longer contribute to better generalization.

3.4.7 Adadelta

In order to address the limitation of Adagrad optimization algorithm, Adadelta helps to put a limitation on the monotonic decrease of the learning rate. In Adagrad, the all the past gradients were used in determining the next update but Adadelta restricted the square of the gradient to a fixed length and uses the decaying exponential of all the accumulated squared gradients, $E_v[g^2]_n$. $E_v[g^2]_t$ depends on previous and the current update and can be expressed as,

$$E_v[g^2]_n = \gamma E_v[g^2]_{n-1} + (1 - \gamma) g_n^2 \tag{3.15}$$

The Adadelta update rule can be expressed as,

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{n-1}}{RMS[\Delta\theta]_n} n \tag{3.16}$$

The new parameter to be estimated can be expressed as,

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{3.17}$$

RMS is the root mean squared error criterion of the gradient. The expression in (3.17) for estimating the next set of parameters is not dependent on the learning rate.

3.4.8 RMSprop

Adadelta and RMSprop are adaptive techniques to improve the Adagrad optimization algorithm. It also aimed at reducing the drastic diminishing learning rate that occurs with Adagrad optimization algorithm capable of stopping the learning process. It has the same derivation steps compared to that of Adadelta. The RMSprop average squared gradient can be expressed as,

$$E_v[g^2]_n = 0.9E_v[g^2]_{n-1} + 0.1g_n^2 \tag{3.18}$$

The new parameter to be estimated can be expressed as,

$$\theta_{n+1} = \theta_n - \frac{\eta}{\sqrt{E_v[g^2]_n) + \in}} \tag{3.19}$$

3.4.9 Adam

The full meaning of Adam is Adaptive Moment Estimation. This also uses adaptive learning rates for each parameter like the AdaDelta and RMSprop optimization algorithms. As earlier discussed while discussing AdaDelta and RMSprop, the decaying exponential of the sum of the squared gradients of both the previous and current updates was considered. One of the ideas used in Adam optimisation algorithm that makes it differs from both AdaDelta and RMSprop optimisation algorithms is the addition of the decaying exponential of the gradients similar to the principle of adding momentum to SGD. It uses the mean (μ_n) and the variance of the gradients (v_n) in its implementation. The mean and the variance are also called the first and second moments of the gradients respectively μ_n and v_n can be expressed as given in eqn. (3.20) and eqn. (3.21) respectively.

$$\mu_n = k_1\mu_{n-1} + (1 - k_2)g_n \tag{3.20}$$

$$v_n = k_1 v_{n-1} + (1 - k_2)g_n^2 \tag{3.21}$$

k_1 and k_2 are constants chosen very close to 1. Unlike AdaDelta and RMSprop, Adam adopts bias correction steps when estimating μ_n and v_n. This is because μ_n and v_n are usually biased towards zero. The Adadelta update rule can be expressed as,

$$\theta_{n+1} = \theta_n - \frac{\eta}{\sqrt{\hat{v}_n + \epsilon}}\widehat{m}_n \tag{3.22}$$

Chapter 4

Stacked Sparse Autoencoder for Source Camera Identification

In this chapter, we proposed the use of deep learning technique called stacked autoencoder to solve the instance-based source camera identification by using PRNU fingerprints. The PRNU fingerprint is proposed as the input data for the stacked autoencoder because it is unique to the individual camera device. We have explained comprehensively in Section 1.2, the motivation for the proposed stacked sparse autoencoder. SSAE has been used successfully for the digit classification since each of the images consists of the same digit value. Therefore, for source camera identification, stacked autoencoder is considered suitable since there is a correlation between PRNU of images of the same camera instance. To the best of our knowledge, we are the first to present the possibility of using deep learning module based on SSAE to solve source camera identification problem. Therefore, we compared our results with some of the states of the art methods for source classification. Using the proposed method, we can achieve good generalization performance without the use of a huge dataset usually used in many deep learning implementations. Therefore, the proposed method is computationally effective.

The rest of Chapter 4 is organized as follows. Section 4.1 introduces the working principle of autoencoders. Section 4.2 describes the architecture of the proposed SSAE for source camera identification. Section 4.3 presents our experimental evaluation on Dresden database, which includes, the datasets description, parameters selection for SSAE, results analysis and comparative study with four states of the art methods. Section 4.4 summarises the work and its contributions.

4.1 Working Principles of Autoencoders

AE is a neural network that has an auto-associative architecture [20, 34]. Its training is done in an unsupervised manner where input data can be trained without the knowledge of the data labels. In this case, the training samples are not labeled and the expected values at the output are same as the input values. Backpropagation technique is adopted to adjust the weights of the network. Given x as an input, an autoencoder output $\hat{x}$ is approximate of the input. Figure 4.1 shows an example of an autoencoder that learns a function,$\hat{x}$ as the as an approximate of the input. It consists of three layers: input, hidden and output layers. The circles with label "+1" are bias units while the circles with label x_i are the inputs to the autoencoder. There are three inputs and one intercept term. The circles with label $\hat{x}_i$ represent the output in which the output $\hat{x}_i$is similar to x_i. The autoencoder attempts to learn a function $h_{w,b}(x)$ such that $h_{w,b}(x) \approx x$. W represents the weight of the network and b is the bias. In this case, the function $h_{w,b}(x)$ is trained to give a compressed form of the input as the number of hidden units is lesser than the number of input units.

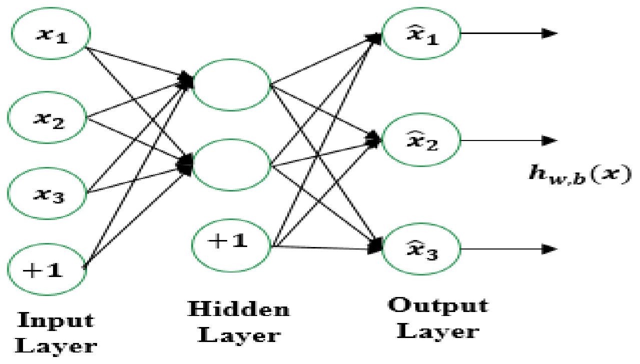


Figure 4.1. An autoencoder.

Given AE training examples to be $\{x_1, x_2, x_3, x_4, \ldots.\}$, then $x_i = y_i$ where, $x_i \in \mathbb{R}^n$, y_i is the output values and $y_i \in \{1,2,3 \ldots . c\}$, c is the number of target outputs. If the number of the hidden units is smaller than a total number of units in the input layer, a compressed form of the input is learned. In this case, the autoencoder acts like a PCA in which interesting structure in the input is learned by the AE. If the number of hidden units is greater than that of the input units, good representation of data can still be obtained by imposing constraints such as sparsity constraint on the activation units. Through the sparsity constraint, useful activation units are extracted to yield a useful representation of input data. The sparsity constraint acts as if a non–linear mapping is learned from the input data so as to generate invariant features representation having high discriminating power during classification. When a sparsity constraint is imposed on an autoencoder, it is called sparsity autoencoder [34]. The work done in [90] shows that the AE trained using sparsity constraints is more efficient that AE trained without sparsity constraints. With reference to Figure 4.1, let ${A^{(2)}}_j$ be the activation units at the second layer of the network, j be the index of the hidden units in the second layer. The average activation per unit in the layer can be expressed as,

$$\hat{\rho}_j = \frac{1}{M}\sum_{i=1}^{M}[{A^{(2)}}_j \, x_i \,] \tag{4.1}$$

where, M is the total number of units in the input layer and $\mathrm{i} = 1,2, \dots . \mathrm{M}$. The sparsity constraint is imposed by making $\hat{\rho}_{\mathrm{j}} = \rho$. ρ is a small value very close to zero. ρ is called the sparsity parameter. Let P be the penalty term which is used to penalize the huge divergence of $\hat{\rho}_j$ from ρ so that the activations per unit will be very small. The idea of imposing penalty is a way of taking absolute value penalty and this helps to regulate the average numbers of zero in the output representation [97]. In [34],the sparsity constraint is imposed on the loss function by calculating the Kullback –Leibler (KL) divergence between desired sparsity parameter and the actual value. However, in Keras implementation of sparsity constraint, the penalty term is either imposed by adding the absolute values of the true value of a layer into the loss function or by adding the square of the true value of a layer into the loss function. For our proposed method, we imposed penalty term (P) by adding the absolute values of the true value of a layer into the loss function. Practical implementation of how sparsity constraint is imposed in Keras framework can be learnt in [35]. The sparse activation is enforced by a regularization parameter, β. This is used to enforce the sparsity constraints on the activation units that are active. The overall cost function can be defined as,

$$\mathrm{J}(\mathrm{W}, \mathrm{b}) = \mathrm{L}_{\mathrm{CE}}(\mathrm{x}, \hat{\mathrm{x}}) + \lambda \sum_{i=1}^{N} ({W_i}^{l})^2 + \beta \mathrm{P} \tag{4.2}$$

where, $\mathrm{L}_{\mathrm{CE}}(\mathrm{x}, \hat{\mathrm{x}})$ is the cross-entropy loss and its equation is given in eqn. (3.6). $\mathrm{L}_{\mathrm{CE}}(\mathrm{x}, \hat{\mathrm{x}})$ measures the error between the output and the learned network. The second term in eqn. (4.2) is the $L2$ regularization or $L2$ norm. The $L2$ regularization helps to reduce overfitting and is imposed on the weight of each encoding layer of the sparse autoencoder. Overfitting occurs when the same model used to train a data performs poorly when evaluated on an unseen data. λ is the weight regularization parameter and W_i^l is the weights of layer l associating with the node i in l from the previous layer. Another novel technique for preventing overfitting of neural networks is the drop-out regularization [98]. Dropout regularization is achieved by randomly selecting a fraction of the units of a layer and set them to zero while the remaining units in the layer will be used as the input to the next layer. Only a fraction of the input neurons is activated [34]. The basic idea of the dropout regularization is to prevent the network from learning redundant features at each hidden layers so as to only retain useful features necessary for better generalization [93]. It also helps to reduce computational demand by only performing activation on useful features. Since we are already learning the compressed representation of our PRNU fingerprint, further reduction of PRNU features using drop-out regularization will lead to a poor generalization for our SSAE. Also, in [98], it was reported that dropout regularization should not be used with deep modules that involve pre-training without any weight constraints or regularization. This is because distinctive features learned either using pre-training or sparsity can be dropped and won't participate in the next layer.

The most suited activation function for a problem is best determined by performing grid search over a range of activation functions. Optimization of the weights and the biases of networks is a very crucial step in the implementation of a sparse autoencoder. Optimization techniques are used for updating hyper-parameters needed for training the DNN. The most recently used optimization algorithms are gradient-based optimization techniques. The work in [96] reviewed some of these techniques which include stochastic gradient descent (SGD), mini-batch gradient descent (MGD), Nesterov accelerated Gradient, Adagrad, Adadelta, and Adam. The most suited optimization algorithm is dependent on the nature of the data and can also be determined by grid search. Furthermore, backpropagation is further used with algorithm optimization techniques to further obtain a well-trained network. The backpropagation helps to propagate the error from the output back to the inputs and weights are updated accordingly [99]. The goal of backward propagation is to further minimize the cost function in eqn. (4.2).

Also, when the input data is corrupted with noise before being given as an input to the auto-encoder, it is called denoising autoencoder (DAE) [100]. The noise corruption aims to ensure that a robust representation of input data that capture its probabilistic distribution can be obtained. In [100], stacked denoising autoencoder (SDAE) was introduced which has a similar working principle with the stacked supervised sparse autoencoder in [71]. SDAE was adopted in [100] for learning useful representations in a deep network with local denoising criterion. In [31] and [101], SDAE was used for the extraction of features for pose-based action recognition and acoustic feature extraction respectively. In this work, SSAE is used to learn robust features since PRNU already has noise characteristics. The network structure of the proposed SSAE is explained in Section 4.2.

4.2 Proposed Stacked Sparse Autoencoder

Figure 4.2 shows the block diagram of our proposed DNN for source camera identification. The PRNU fingerprint is used as the input data (x) for our proposed DNN. For training, robust features of our PRNU features are learned by stacking several hidden layers of an autoencoder. A deep network is formed when the encoding features of the last hidden layer are passed into a softmax classifier. To further minimize the reconstruction error and to obtain a well-trained DNN, all the weights of all the hidden layers in the network as well as the weights of the softmax classifier are fine-tuned using the back propagation discussed in Section 4.1. It is called supervised fine-tuning because the knowledge of the training and the testing targets are needed for classification purpose.

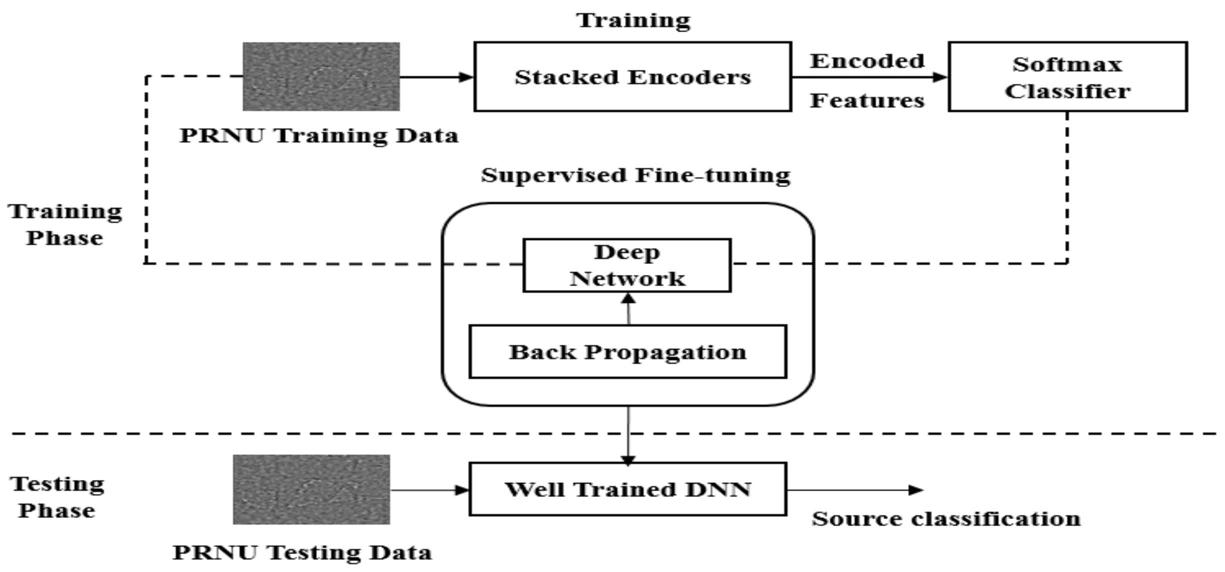


Figure 4.2. Block diagram for the proposed DNN for source camera classification.

After fine-tuning of the deep network, PRNU fingerprints of testing data were fed into the well-trained network to determine their source camera. In the remaining part of this section, we will discuss details of our proposed DNN. This includes the network architecture of the proposed methods, supervised fine-tuning and PRNU extraction and preparation.

4.2.1 Network Architecture of the Proposed Stacked Autoencoder

An autoencoder as earlier explained has input, hidden and output layers. A stacked autoencoder works by extracting the features of the hidden layer (encoding features) of an autoencoder and given these features as input to another autoencoder. The output features (decoding features) of the autoencoder are discarded. The features of the hidden layer are a representation of the input data in another domain. This process is done recursively until the robust representation of the input data is obtained and passed into a supervised classifier for generalization. Figure 4.3 (a) shows the architecture of training stage while Figure 4.3 (b), shows the fine-tuning stage of our proposed SSAE. In Figure 4.3 (a), the first encoding features; $E_1(W, b)$ were extracted and passed as input to another autoencoder. The encoding features of the second autoencoder, $E_2(W, b)$ were extracted and fed in as input to the next autoencoder. Finally, the encoding features , $E_N(W, b)$ were then extracted as the trained features. $E_N(W, b)$ is the encoding features of Nth hidden layer of the last autoencoder, N is the hidden layer of the last autoencoder where optimal features are obtained during the training process. As shown in Figure 4.3 (b), the optimal encoded features are fed as input into the softmax classifier to form a deep network. To have a well-trained deep network, a supervised fine-tuning is performed. In other words, all the weights at the hidden layers and the softmax classifier are fine-tuned to further reduce training error by using the back-error propagation [102]. The optimal features are obtained by passing each encoding features directly for supervised fine-tuning to determine the Nth hidden layer whose encoding features give the optimal classification accuracy.

A logistic regression classifier is a linear classifier that determines the class of a target output based on the probabilistic prediction. It is used for binary classification. To generalize for multiple classes, Softmax classifier can be used [21]. Figure 4.4 shows a softmax model. The dimension of an input x is N and the number of classes is c. Let $n = 1{:}N$, $j = 1{:}c$, and i represents each column vector in an input data x. In logistic regression, each input vector corresponding to a class is projected into a set hyperplanes. The probability that an input belongs to a class j is determined by calculating the distance between the inputs and the hyperplane. Given that the target output is denoted as Y, then the probability that an output, y, belongs to target class i when parameterized by weights W and biases b can be expressed as,

$$\mathrm{P}\left(\mathrm{Y} = \mathrm{j} \middle| \mathrm{x}^{\mathrm{i}}, \mathrm{W}, \mathrm{b}\right) = \mathrm{softmax}_{\mathrm{j}}\left(\mathrm{Wx}^{\mathrm{i}} + \mathrm{b}\right) = \frac{\mathrm{e}^{\mathrm{Wx}^{\mathrm{i}}+\mathrm{b}}}{\sum_{\mathrm{j}} \mathrm{e}^{\mathrm{Wx}^{\mathrm{i}}+\mathrm{b}}} \tag{4.3}$$

The expression in (4.3) gives the probabilities of all classes given an output Y. The predicted output $(\mathrm{y_p})$ is the class that produces the maximum probability. This can be expressed as,

$$\mathrm{y_p} = \underset{\mathrm{j}}{\mathrm{armax}}\, \mathrm{P}\left(\mathrm{Y} = \mathrm{j} \middle| \mathrm{x}^{\mathrm{i}}, \mathrm{W}, \mathrm{b}\right) \tag{4.4}$$

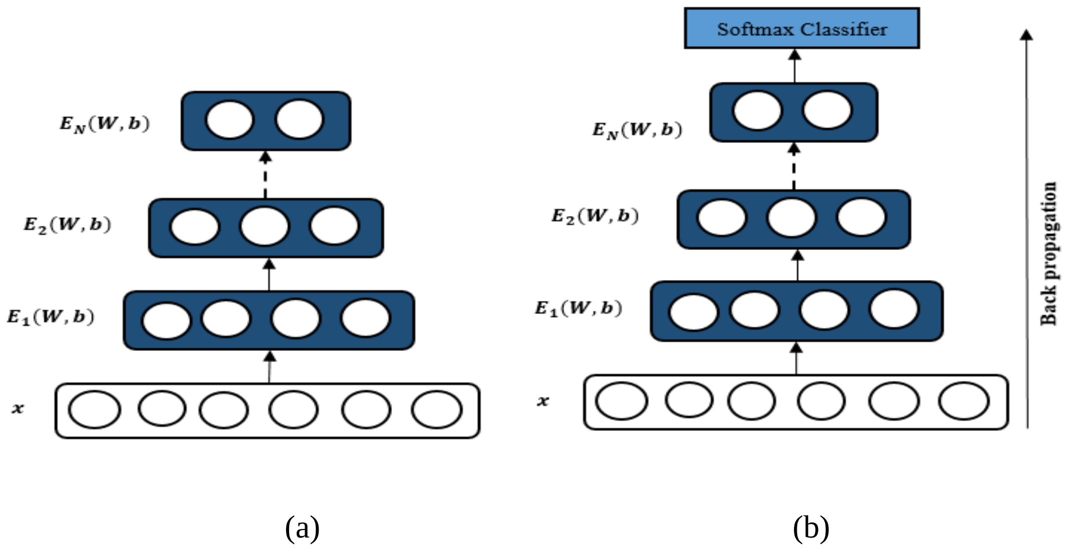
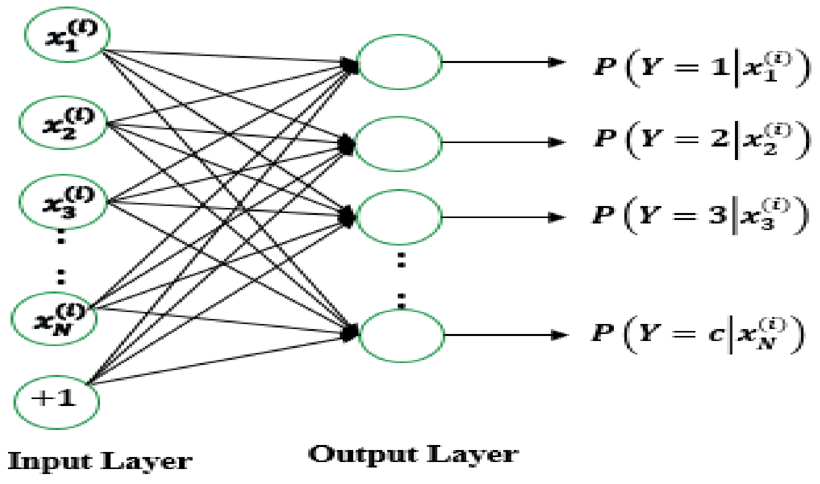


Figure 4.3. (a) Training stage (b) fine-tuning stage in SSAE.



$x_1^{(i)}$
$x_2^{(i)}$
$x_3^{(i)}$
$x_N^{(i)}$
+1
$P\left(Y=1\middle|x_1^{(i)}\right)$
$P\left(Y=2\middle|x_2^{(i)}\right)$
$P\left(Y=3\middle|x_3^{(i)}\right)$
$P\left(Y=c\middle|x_N^{(i)}\right)$
Input Layer
Output Layer

Figure 4.4. A softmax model.

The source identification was achieved by feeding the testing data into the fully and well trained DNN. Besides, the progressive decrease in the numbers of the hidden units in each layer shows that the compressed representation of the PRNU input was learned.

4.2.2 PRNU Extraction and Preparation

In this work, we adopt instance-based source camera identification, instead of using the original images in our database, we propose to use PRNU as the input data since autoencoders are dataset-specific. Original image contains not only sensor noise information but also image scene details. It is difficult to characterize structures in the sensor noise in the presence of image details. The sensor noise PRNU is unique to each individual camera. The PRNU can be extracted by subtracting its denoised version from the original image. The denoising filter proposed by Lukas et al. [7] was used for denoising. To further enhance the PRNU fingerprint against in-camera pre-processing artifacts, both training and testing PRNU fingerprints are preprocessed by the zero-mean operation. Zero-mean operation implies to find the mean of each row or column and then subtract the mean from each element in the rows and columns respectively [47]. All the PRNU fingerprints after the zero-mean operation are transformed into the column-wise vector. No further transformation techniques were performed on the PRNU except the zero-mean operation. Given that the resolution of the images is $p \times p$, then the dimension of PRNU signal is $p^2 \times 1$. If n images are used per camera, the first n columns are the fingerprints of the first camera. The following n columns are for the next camera and so on. If N cameras are to be used, the total number of columns in the data is nN. Our PRNU data can be arranged as a matrix of size $p^2 \times nN$. Given that n images are used per camera, then, the first n columns are the fingerprints of the first camera. The next n columns are for the next camera and so on. If N cameras are to be used, the total number of columns in the data can be given as $n \times N$. Our PRNU data will be a matrix of size $p^2 \times nN$. It should be noted that it is not necessary that the same numbers of images must be used for each camera. When the same samples are used per class during training, it is called balance classification problem. Otherwise it is called imbalance classification problem [33]. In forming the labels of the PRNU data, a column vector of size $N \times 1$ is generated. The column vector consists of only one non-zero element whose position indicates the source camera of the PRNU fingerprint.

The PRNU fingerprint has noise-like characteristics and its domain is different from features directly extracted from the images (data-driven approach) like most of the object recognition problems. Therefore, direct application of the existing parameters successfully applied to those problems won't yield best network generalization. Therefore, obtaining optimal parameters suitable to learn a robust or useful representation of our PRNU fingerprints will also be dependent on the background knowledge of PRNU signal and searching over a range of parameters (grid-search technique). This is essential in choosing the weight initialization technique.

4. 3 Experiments and Results

4.3.1 Experimental Settings

The experiments were carried out using photos from the Dresden database [103]. Our choice of selecting images of cameras for our experiments from the Dresden database is because it is an image database specifically built for development and benchmarking of camera-based digital forensic techniques. Another database usually used for camera attribution is the Flickr database. Flickr database is only suitable for camera model identification and not camera instance or source camera identification. This is because people share images of different devices of the same model and there is no distinction on the exact device that captured the images in the Flickr database. Therefore, Dresden database is suitable for experimental analysis of source camera identification since images are classified per camera instances. Table 4.1 shows the lists of natural images of 20 cameras used in the evaluation of the proposed method. The camera band "Kodak_M1063_0" means that the photos are taken by a device whose manufacturer is Kodak and camera model is "M1063" and "0" is the device identification number. Kodak_M1063_1 simply means another device of Kodak_M1063.

PRNU of images from the same camera devices is only the same if they are extracted from the same spatial location of the images. Therefore, single patch from each image of cameras at the same spatial location is used for estimation of noise residues. In most conventional methods, the use of matching of the reference PRNU images of cameras with the PRNU of testing images of cameras, involve the use of vector operations. Therefore, the use of original image size becomes complex and highly computationally expensive during the matching process [104]. In our proposed methods, our focus is source camera identification of small size images and applications involving splicing translocation. Besides, the use of deep learning methods involves the use of optimisation algorithms and the use of high feature dimensions will result in infinite iterations and retries. This usually slow down the rate of convergence to global minimum and hence, results in poor generalization accuracy. All images are centered cropped at resolutions; 64×64, 128×128 and 256×256. Centre patches of images are used because not all parts of the images are rich in PRNU fingerprints and images with dark regions have weak PRNU. Saturated pixels cause undesirable noise in residual signals and center patch contains PRNU with homogenous features. Furthermore, as shown in the work in [105], there are some of the images with vignetting effect. Vignetting effect occurs when edges of an image are darker than the center of the image. Therefore, using the center patch also helps to reduce the vignetting effect and hence improves the quality of the extracted PRNU images [105]. To properly evaluate the SSAE and ensure a large amount of training data are used, images are divided into training and testing sets. The division steps are as follows:

- The natural images are divided by random splitting into sets A and B respectively. Set A consist of 80% of the natural images while Set B consists of the remaining 20%.
- Set A is used as training images while Set B is used as the testing images.

All the experiments were carried out on a computer with 4.00GHz Intel (R) Core (TM) i7-6700k CPU and 1 terabyte memory. The PRNU feature extraction of all image was carried out using MATLAB 8.6.0 on window 10. The SAE was carried out on window 10 with Keras using Theano backend on a 11 G GPU NVidia GTX 1080Ti. Keras with Theano backend is a deep learning framework. It is known for its simplicity and ease of use. Keras library was used with the scikit-learn library to leverage the power gain of SAE evaluation and optimization of hyperparameters. Both Keras and scikit-learn are frameworks based on Python programming.

4.3.2 Parameters Selection for SSAE

We will discuss initialization of the weights of neural networks for PRNU fingerprint before explaining parameters selection for SSAE. The weights initialization are selected in a way as to achieve global convergence. The closer they are to a global minimum, the better. Though global convergence is also dependent on the training algorithm. There are several methods for setting initial weights of NN. Setting initial weights for neural networks has no standard formula. Different generalization results are usually obtained depending on the domain of adaptation. Though random or normal initializations are usually used for image classification problems [65, 106, 107], however, initialization depends on the feature scaling or normalization technique carried out on the data. The structure of the PRNU is best exposed to the neural network for learning after zero-mean operation has been applied. Most of the other normalization techniques tested on the PRNU gave us poor results. In [65], data must be normalized before using their proposed weight initialization. In our work, using zeros or ones as weights initialization are most suited for PRNU fingerprints with SSAE. Using ones or zeros as NN weight initialization initializes the NN to start with uniform distribution. This enables the NN to learn its weights till it obtains optimum values. The third method described in [108] used zero weight initialization for the multi-layer neural network. As earlier explained, compressed representation of activated neurons is learned at each hidden layer of our proposed network. Therefore, the problem of breaking the symmetry of networks using random or normal weight initializations is not a challenge since the signals (hidden units) at hidden layers of the structure of our proposed SSAE are not the same. $L2$ weight and sparsity regularizations are also applied to the weights of our networks to avoid overfitting.

Table 4.1. Details about the cameras used in the experiment including the resolution and number of images.

S/N	Camera Brand	Resolution	Number of Natural Images
1	Agfa_DC-830i	3264 × 2448	342
2	Olympus_mju_1050SW_0	3264 × 2736	204
3	Olympus_mju_1050SW_1	3264 × 2736	209
4	Kodak_M1063_0	3664 × 2748	464
5	Kodak_M1063_1	3664 × 2748	458
6	Agfa_Sensor530s_0	4032 × 3024	372
7	Nikon_D70_0	3008 × 2000	180
8	Nikon_D70_1	3008 × 2000	189
9	Olympus_mju_1050SW_2	3264 × 2736	218
10	Canon Ixuss 55	2592 × 1944	224
11	Canon_Ixus70_0	2304 × 3072	171
12	Canon_Ixus70_1	2304 × 3072	179
13	Canon_Ixus70_2	2304 × 3072	171
14	Samsung_L74wide_0	2304 × 3072	229
15	Samsung_L74wide_1	2304 × 3072	224
16	Samsung_L74wide_2	2304 × 3072	231
17	Samsung_NV15_0	2304 × 3072	217
18	Samsung_NV15_1	2304 × 3072	214
19	Sony_DSC-H50_0	2736 × 3648	266
20	Sony_DSC-H50_1	2736 × 3648	234
Total number of Images			**4996**

Network configuration parameters and training parameters can be sub-divided into two categories. We have the model and fine-tuning parameters. The key training parameters include the numbers of hidden layers of SAE, the number of the units in the hidden layers, weight and sparsity regularization parameters. The key fine-tuning parameters include learning rate and the number of epochs. The commonly used searching techniques for selecting optimal network parameters both for the training and fine-tuning stages are usually the grid search or random search. The optimal parameters cannot be determined at once and therefore; the best approach is to fix some parameters while we use a grid search to determine the other optimal parameters. In our experiments, we geometrically determined the number of the hidden units per layer. For example, using 64×64, the total number of pixels per column becomes $4096 = 64 \times 64 = 64 \times 8^2$ pixels. The number of hidden units is geometrically decreased per hidden layer. The number of neurons per lth hidden layer is defined as $64 \times (8 - l)^2$. Similarly, for 128 ×128, the total number of pixels per column becomes $16384 = 128 \times 128 = 4 \times 64^2$. The number of neurons per lth hidden layer is defined as $4 \times (64 - l)^2$. Hidden units were also selected similarly for 256×256. Though, there is no standard way of determining hidden units per layer (h_l) but given that $l = 1, 2 \ldots N$, the parameters can also be tuned within $h_l \; and \; h_N$.

We searched the best results for our stacked autoencoder given $that \;\; l = \; 1, \; 2, 3..N.$ N is the hidden layer with the optimal performance. The sparsity and weight regularization parameters are fixed at 0.00001. We will show how to determine the optimal number of epoch, the hidden layers, learning rate (η), activation function and the optimization algorithm most suitable for generalization by grid search. Firstly, we determined the optimal hidden layers by fixing number a of epoch, the the activation the function and optimizer as 20, ReLU and SGD ($\eta = 0.001$) respectively. Table 4.2 shows the overall identification accuracies of cameras at different image resolutions for four hidden layers using the first 10 cameras in Table 4.1. By overall accuracy, we mean the ratio of the all correctly identified samples of all the cameras to the total number of testing samples of all the cameras. It was observed from Table 4.2 that the optimal performances for 64×64 and 128×128 were obtained at hidden layer 2 while it is at layer 3 for 256×256. The bold numbers in Table 4.2 show the optimal performance for each image size.

Table 4.2. Overall Identification accuracies at each hidden layer (%).

Image Size	Average Accuracy after fine-tuning			
	Layer 1	Layer 2	Layer 3	Layer 4
64×64	46.85	**47.90**	44.93	43.18
128 × 128	68.53	**69.58**	69.23	66.43
256 × 256	86.19	86.63	**87.76**	87.13

Having established the optimal layer for our SAE for the different image sizes, we then obtained the best epoch with global convergence using grid search within; [10, 20, 50, 70,100]. Table 4.3 shows the overall average accuracies of cameras after fine-tuning for the range of epochs used. In Table 4.3, it was observed that the best overall accuracies for 64 × 64, 128×128 image sizes after fine-tuning are obtained when the number of the epochs is set to 10 while it is at 50 for 256 × 256. The bold numbers in Table 4.3 show the optimal performance for ranges of epochs used. As observed in Table 4.3, there is difficulty in achieving global convergence using SSAE with improved identification accuracies especially for an image size of 64 × 64 while there is improved convergence for image sizes 128 × 128 and 256 ×256 and this reflected in their identification accuracies of 70.63% and 87.76% respectively compared to 47.90% identification accuracy for 64 × 64 image size. The higher the image size, the better the quality of PRNU images and hence, the better the rate of global convergence. The rate of convergence also depends on the generalization capability of the applied model. One of the key parameters for fine-tuning as earlier stated is the learning rate. We further carried out experiments to check how the performance varies for different learning rates. The learning rates used are 0.2, 0.25, 0.02, 0.001:0.005. The overall accuracies after fine-tuning at different learning rates are shown in Figure 4.5.

Figure 4.5 shows 0.001 as the learning rate at which best overall accuracies are obtained for all the images sizes used.

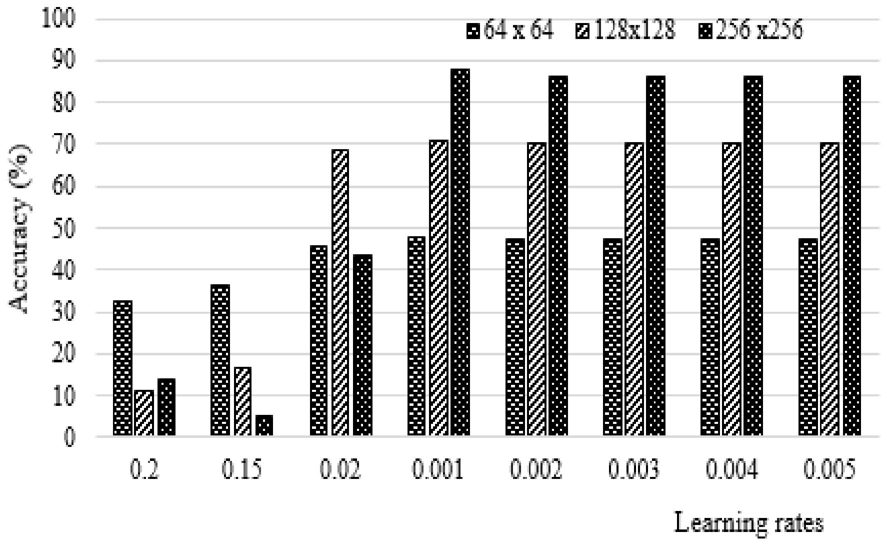


Figure 4.5. Overall accuracies after further fine-tuning at different learning rates.

Table 4.3. Optimal results for each epoch for the best-hidden layer (%).

Image Size	**Number of Epochs**				
	10	20	50	70	100
64 × 64	**47.90**	**47.90**	46.68	**46.50**	45.98
128 × 128	**70.63**	69.23	68.01	68.36	67.83
256 × 256	87.41	**87.76**	**87.76**	87.59	87.24

4.3.3 Experimental Results

Experiment 1

We carried out an experiment on SSAE for source camera identification on the first ten cameras as shown in Table 4.1. We refer to this experiment as case 1. The parameters selection and intermediate results have been discussed in Section 4.3.2. In this section, we summarize and discuss the identification accuracies of the 10 cameras. The summary of the overall accuracies for all the image sizes after fine-tuning is shown in Table 4.4. Table 4.4 shows that the overall accuracies after fine-tuning the proposed SSAE are 47.90%, 70.63% and 87.76% for 64×64, 128×128 and 256×256 respectively. It was observed that the higher the image size, the better the camera identification accuracies. This can be attributed to the fact that, the higher the image resolution, the stronger the PRNU fingerprint. Better or stronger PRNU fingerprint means quality data is being used as the input for SSAE. Apart from using large dataset for generalization of deep learning modules, one of the factors that have a significant contribution to generalization is the quality of information in the data.

The maximum image size used in our experiment is 256×256 and this means 65536 elements per column for one image of a camera. The confusion matrixes obtained using our proposed method are shown in Table 4.5, Table 4.6 and Table 4.7 for 64×64, 128×128 and 256×256 respectively. The bold numbers in Table 4.5, Table 4.6 and Table 4.7 indicate the identification accuracy for each camera device. Nikon_D70_0 and Nikon_D70_1 have poor identification accuracies of 15.79% and 8.11% respectively for 64×64. The accuracies have a significant improvement to 35.58% and 53.35% for 128×128 respectively. Finally, Nikon_D70_0 and Nikon_D70_1 for 256×256 are 76.32% and 83.78% respectively. The performance of the two devices might improve if the training labels for the cameras are increased. The poor performance can also be attributed to the Nikon cameras having weak PRNU.

Table 4.4. Overall identification accuracies at different image sizes (%).

64×64	128×128	256×256
47.90	**70.63**	**87.76**

Experiment 2

We further investigated on how the proposed SAE generalizes to a new set of camera devices using the same optimal hyperparameters, activation function, and optimizer in Section 4.3.2. This was carried out using the remaining 10 cameras in Table 4.1. We refer to this experiment as case 2. Table 4.8 shows the identification accuracies for the new set of ten cameras for 64×64, 128×128 and 256×256. The overall accuracies for the cameras are 63.19%, 87.72% and 92.28% for 64×64, 128×128 and 256×256 respectively. The results show that hyperparameters, optimizer and activation function used for the SSAE in case 1 is not data specific. The better performance of the cameras in case 2 compared to case 1 can be attributed to some of the cameras in case 2 having stronger PRNU than those in case 1. The experiment further shows that there is no need to carry out any further search to obtain best optimal parameters for the PRNU of the new set of cameras. The implication of this is that there is no further computational time incurred due to the grid – search or manual search for optimal parameters.

Experiment 3

In this experiment, we investigated the generalization capability of our proposed method by using all the 20 cameras in Table 4.1. We refer to this experiment as case 3. All that is required is to concatenate the already extracted PRNU of the cameras in case 1 and case 2 respectively. Also, the target labels must correspond to the each PRNU of cameras. Table 4.9 shows the identification accuracies of cameras for 64×64, 128×128 and 256×256 respectively. Average accuracy per camera device is 37.24% 67.35% and 91.67% for 64×64, 128×128 and 256×256 respectively.

The average identification accuracy per camera for each resolution using our proposed method has a slight decrease when compared to identification accuracies using only 10 cameras as of case 1 and case 2. In machine learning, the more the number of targets to be classified, the lower the classification accuracies of each class. Our experimental results show that our proposed method still has significant performance using 128×128 and 256×256 image size for the 20 cameras 128×128 and 256×256 respectively.

Visualization of the input and extracted optimal features for 64×64, 128×128 and 256×256 are shown in Figure 4.6, Figure 4.7 and Figure 4.8 respectively.

Table 4.5. Identification accuracy (in percentage points %) of the proposed method for 64 × 64 images size (case 1).

Camera Device		1	2	3	4	5	6	7	8	9	10
Agfa_DC-830i	1	**73.24**	1.41	1.41	7.04	2.82	9.86	1.41	-	1.41	1.41
Olympus_mju_1050SW_0	2	7.14	**26.19**	11.90	9.52	19.05	11.90	-	-	14.29	-
Olympus_mju_1050SW_1	3	9.30	2.33	**27.9**1	4.65	18.60	11.63	-	2.33	18.60	4.65
Kodak_M1063_0	4	6.49	-	3.90	**45.45**	16.88	9.09	5.19	2.60	7.79	2.60
Kodak_M1063_1	5	7.14	2.38	1.19	8.33	**54.76**	9.52	1.19	4.76	3.57	7.14
Agfa_Sensor530s_0	6	5.41	1.35	-	6.76	6.76	**74.32**	1.35	0.00	1.35	2.70
Nikon_D70_0	7	18.42	5.26	-	15.79	15.79	15.79	**15.79**	2.63	10.53	-
Nikon_D70_1	8	13.51	5.41	5.41	18.92	16.22	10.81	-	**8.11**	5.41	16.22
Olympus_mju_1050SW_2	9	6.38	8.51	8.51	4.26	14.89	6.38	-	2.13	**42.55**	6.38
Canon Ixuss 55	10	1.69	3.39	8.47	1.69	8.47	11.86	1.69	0.00	5.08	**57.63**

Table 4.6. Identification accuracy (in percentage points %) of the proposed method for 128 × 128 images size (case 1).

Camera Device		1	2	3	4	5	6	7	8	9	10
Agfa_DC-830i	1	**97.18**	-	-	-	-	1.41	1.41	-	-	-
Olympus_mju_1050SW_0	2	4.76	**52.38**	-	11.90	11.90	7.14	-	-	11.90	-
Olympus_mju_1050SW_1	3	13.95	9.30	**48.84**	2.33	6.98	6.98	-	-	-	11.63
Kodak_M1063_0	4	9.09	3.90	3.90	**63.64**	10.39	1.30	2.60	2.60	1.30	1.30
Kodak_M1063_1	5	2.38	3.57	1.19	10.71	**73.81**	-	2.38	1.19	2.38	2.38
Agfa_Sensor530s_0	6	2.70	1.35	1.35	2.70	2.70	**89.19**	-	-	-	-
Nikon_D70_0	7	10.53	-	7.89	-	12.16	13.16	**35.58**	12.16	7.89	2.63
Nikon_D70_1	8	15.22	2.70	-	10.81	7.11	2.70	2.70	**53.35**	2.70	2.70
Olympus_mju_1050SW_2	9	4.26	2.13	4.26	2.13	2.13	2.13	4.26	2.13	**74.47**	2.13
Canon Ixuss 55	10	-	3.39	1.69	3.39	1.69	3.39	1.69	-	1.69	**83.05**

Table 4.7. Identification accuracy (in percentage points %) of the proposed method for 256 × 256 images size (case 1).

Camera Device		1	2	3	4	5	6	7	8	9	10
Agfa_DC-830i	1	98.59	-	-	-	1.41	-	-	-	-	-
Olympus_mju_1050SW_0	2	-	83.33	-	4.76	4.76	2.38	2.38	-	2.38	-
Olympus_mju_1050SW_1	3	-	6.98	72.09	4.65	4.65	-	4.65	2.33	2.33	2.33
Kodak_M1063_0	4	1.30	2.60	-	84.42	3.90	3.90	1.30	2.60	-	-
Kodak_M1063_1	5	-	-	-	10.71	82.14	3.57	-	2.38	-	1.19
Agfa_Sensor530s_0	6	-	-	-	0.75	-	98.50	0.00	0.75	-	-
Nikon_D70_0	7	2.63	-	-	5.26	5.26	2.63	76.32	7.89	-	-
Nikon_D70_1	8	2.70	2.70	-	2.70	5.41	-	2.70	83.78	-	-
Olympus_mju_1050SW_2	9	-	4.26	2.13	2.13	-	-	-	2.13	89.36	-
Canon Ixuss 55	10	-	1.69	-	-	-	-	-	-	-	98.31

Table 4.8. Identification accuracy (in percentage %) of proposed method (case 2).

S/N	**Camera Brand**	**64 × 64**	**128 × 128**	**256 × 256**
11	Canon_Ixus70_0	39.50	86.80	97.40
12	Canon_Ixus70_1	38.90	77.80	94.40
13	Canon_Ixus70_2	37.50	90.60	100.00
14	Samsung_L74wide_0	40.00	61.90	93.30
15	Samsung_L74wide_1	26.20	70.70	92.90
16	Samsung_L74wide_2	29.30	89.70	95.10
17	Samsung_NV15_0	46.20	88.00	100.00
18	Samsung_NV15_1	52.00	86.00	100.00
19	Sony_DSC-H50_0	93.80	100.00	100.00
20	Sony_DSC-H50_1	85.40	97.60	100.00
Average		**49.92**	**84.70**	**97.30**

Table 4.9. Identification accuracy (in percentage %) of the proposed method (case 3).

Camera Brand	**64 × 64**	**128× 128**	**256× 256**
Agfa_DC-830i	64.44	96.60	96.6
Olympus_mju_1050SW_0	24.40	48.80	80.50
Olympus_mju1050SW_1	10.40	41.7	79.20
Kodak_M1063_0	52.22	61.10	87.80
Kodak_M1063_1	46.10	70.80	83.10
Agfa_Sensor530s_0	62.70	39.00	92.80
Nikon_D70_0	9.80	83.10	86.50
Nikon_D70_1	18.90	54.10	82.10
Olympus_mju_1050SW_2	46.20	69.20	100.00
Canon Ixuss 55	27.30	79.50	90.60
Canon_Ixus70_0	25.00	78.10	87.80
Canon_Ixus70_1	31.70	63.40	96.90
Canon_Ixus70_2	34.44	65.60	94.10
Samsung_L74wide_0	39.20	52.90	94.20
Samsung_L74wide_1	13.50	50.00	96.20
Samsung_L74wide_2	29.40	61.80	97.00
Samsung_NV15_0	32.00	66.00	94.00
Samsung_NV15_1	42.40	78.80	93.90
Sony_DSC-H50_0	69.20	92.30	100.00
Sony_DSC-H50_1	65.40	94.20	100.00
Average	**37.24**	**67.35**	**91.67**

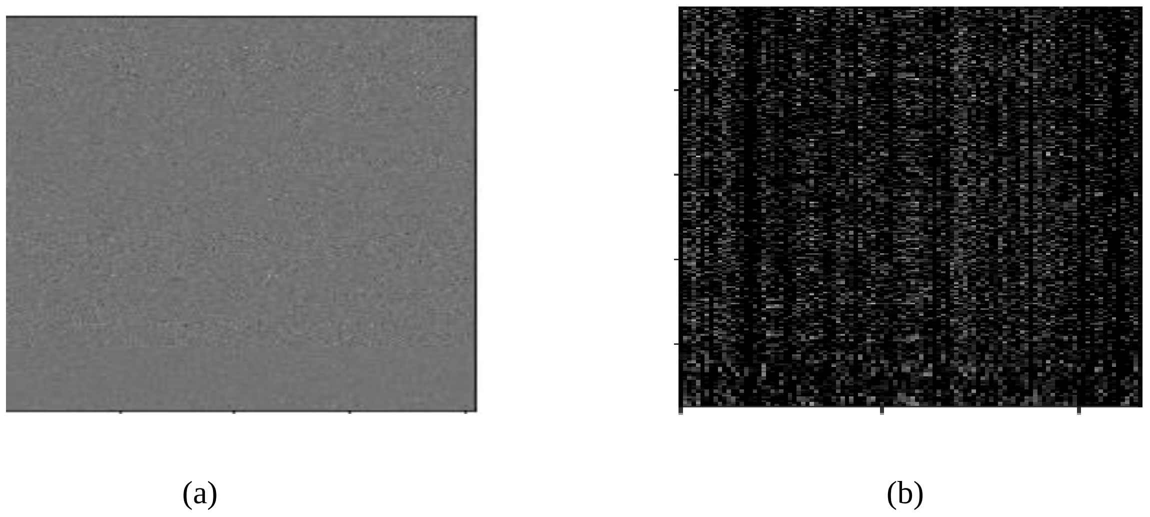


Figure 4.6. Input features (a) and optimal features (b) of SAE for 64 × 6

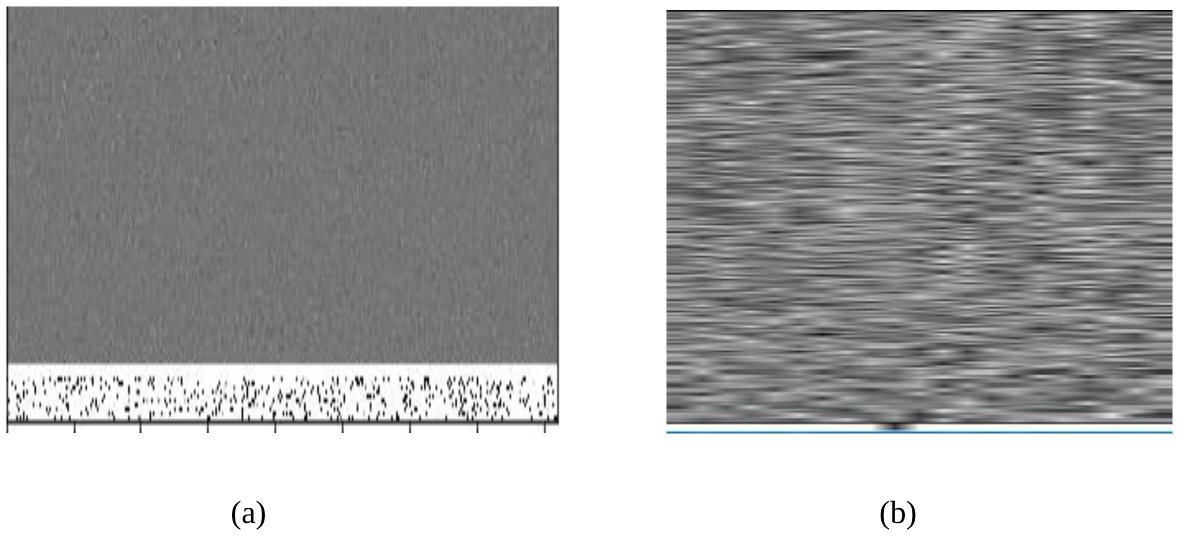
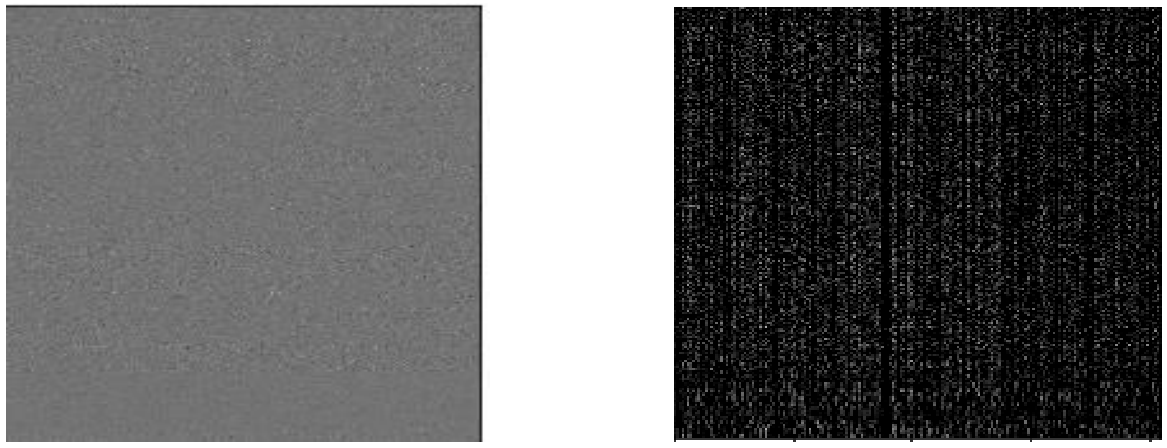


Figure 4.7. Input features (a) and optimal features (b) of SAE for 128 × 128



(a) (b)

Figure 4.8. Input features (a) and optimal features (b) of SAE for 256 × 256

4.3.4 Comparison with Some State-of-the-Art Methods

We compared our proposed deep network with some state-of-the-art methods. The overall identification accuracy for the 20 cameras using our proposed method for the 64 × 64 is very low but considerably high for 128 × 128 and 256 × 256 as earlier explained in experiment 3 in Section 4.3.3. In this section, we compared our proposed method with four states of the art methods: MLE SPN [8], Phase SPN [10], the Li's model [9] and weighted averaging (WA) [51] using 128 × 128 and 256× 256 images sizes. We used Li's model 5 because it gives good performance compared to other models. For all the methods, the classification was done using peak to correlation energy (PCE) [10].

Experiment 1

In this experiment, we compared our proposed methods with these methods using the same setting used for the proposed SAE. This implies that 80% of the images of each camera in Table 4.1 were used for the extraction of the PRNU fingerprints while the remaining 20% were used for testing.

We refer to this experiment as case 4. Table 4.11 and Table 4.10 shows the identification accuracies for 20 cameras of the Dresden image dataset for 128 × 128 and 256 × 256 image sizes respectively. The overall average accuracies using 128 × 128 are 70.15%, 70.16%, 75.08%, 64.16% and 67.09% for MLE SPN, Li's model 5, Phase SPN, WA and SAE respectively. The overall average accuracies using 256 × 256 are 91.71%, 91.90%, 92.73%, 92.38% and 91.67% for MLE SPN, LI's model 5, Phase SPN, WA, and SAE respectively. The identification of our proposed method has comparable detection accuracy with the existing methods though higher identification accuracy than WA for 256 × 256 image size. The camera identification accuracy of the proposed SSAE only has comparable results with the state-of-the-art methods on Dresden database using natural images for training. To adopt SSAE, we must convert the image to one-dimensional features and hence SSAE could not account for spatial information between the pixels of images. Therefore, this negatively impacts the generalization capacity of the SSAE for source camera identification. SSAE is more effective for data with high correlation. Since we used individual PRNU images of cameras, single noise residue representing image will have scene contamination and hence finding ways of improving the quality of the PRNU could be a better way of having highly correlated data which will be more suitable for learning using SSAE. Though, there are camera cases where our proposed method has higher identification accuracy than the all other methods. Another advantage of the proposed method unlike several deep learning techniques is that good performance can be achieved using considerably smaller dataset unlike huge dataset usually required for good performance for deep learning techniques such as CNN. This furthers reduces computational demand for the implementation of the proposed method.

Experiment 2

In all the experiments carried out so far, our proposed stacked autoencoder was trained using natural images of the Dresden database in Table 4.1. There are a limited number of flat images (50 images per camera) and hence not suitable for training deep network. Another reason for not using flat images is that flat images are rarely available for practical purposes or cases involving digital investigation. However, quality PRNU can be best extracted from using flat images unlike natural images with more texture complexity.

Table 4.10. Identification accuracies (in percentage %) compared to other methods for 256 × 256 (case 4).

S/N	**Camera Brand**	**MLE**	**Li's Model 5**	**Phase SPN**	**WA**	**Proposed method**
1	Agfa_DC-830i	92.75	91.30	92.75	94.20	96.6
2	Olympus_mju_1050SW_0	100.00	95.12	95.12	97.56	80.50
3	Olympus_mju_1050SW_1	95.23	95.24	95.24	100. 0	79.20
4	Kodak_M1063_0	78.45	75.27	75.27	72.04	87.80
5	Kodak_M1063_1	73.91	68.48	71.74	69.57	83.10
6	Agfa_Sensor530s	100.00	100.00	100.00	100.00	92.80
7	Nikon_D70_0	61.11	61.11	63.89	63.15	86.50
8	Nikon_D70_1	65.79	71.05	76.32	76.32	82.10
9	Olympus_mju_1050SW_2	97.72	97.72	97.72	97.72	100.00
10	Canon Ixuss 55	100.00	100.00	100.00	100.00	90.60
11	Canon_Ixus70_0	100.00	100.00	100.00	100.00	87.80
12	Canon_Ixus70_1	100.00	100.00	100.00	100.00	96.90
13	Canon_Ixus70_2	100.00	100.00	100.00	94.44	94.10
14	Samsung_L74wide_0	93.48	95.65	97.82	97.83	94.20
15	Samsung_L74wide_1	88.89	97.78	97.78	91.49	96.20
16	Samsung_L74wide_2	93.62	91.49	95.74	95.56	97.00
17	Samsung_NV15_0	97.72	100.00	97.72	97.72	94.00
18	Samsung_NV15_1	95.35	100.00	97.67	100.00	93.90
19	Sony_DSC-H50_0	100.00	100.00	100.00	100.00	100.00
20	Sony_DSC-H50_1	100.00	100.00	100.00	100.00	100.00
Average		**91.71**	**91.90**	**92.73**	**92.38**	**91.67**

Table 4.11. Identification accuracies (in percentage %) compared to other methods for 128 × 128 (case 4).

S/N	**Camera Brand**	**MLE**	**Li's Model 5**	**Phase SPN**	**WA**	**Proposed method**
1	Agfa_DC-830i	84.06	84.06	84.06	88.44	93.20
2	Olympus_mju_1050SW_0	58.54	60.98	63.41	48.78	48.80
3	Olympus_mju_1050SW_1	69.05	64.29	73.81	71.43	37.50
4	Kodak_M1063_0	54.84	49.46	43.01	27.96	55.60
5	Kodak_M1063_1	45.65	45.65	50.00	52.17	69.70
6	Agfa_Sensor530s	89.33	90.67	97.33	10.66	84.30
7	Nikon_D70_0	25.00	25.00	38.89	30.56	22.00
8	Nikon_D70_1	36.84	42.10	42.10	39.47	40.50
9	Olympus_mju_1050SW_2	72.72	79.54	88.63	54.54	71.80
10	Canon Ixuss 55	82.22	77.78	86.67	88.88	86.40
11	Canon_Ixus70_0	91.43	85.71	91.43	85.71	78.10
12	Canon_Ixus70_1	86.11	83.33	83.33	77.77	61.00
13	Canon_Ixus70_2	85.71	91.43	82.85	85.71	78.10
14	Samsung_L74wide_0	65.22	63.04	71.34	60.87	51.00
15	Samsung_L74wide_1	51.11	48.89	71.11	53.33	50.00
16	Samsung_L74wide_2	44.68	59.57	68.09	57.45	64.70
17	Samsung_NV15_0	84.09	86.38	79.54	75.00	72.00
18	Samsung_NV15_1	76.60	69.39	86.04	74.42	84.80
19	Sony_DSC-H50_0	100.00	100.00	100.00	100.00	94.20
20	Sony_DSC-H50_1	100.00	95.74	100.00	100.00	98.10
Average accuracy per camera		**70.16**	**70.15**	**75.08**	**64.16**	**67.09**

In this experiment, the capacity of PRNU to increase the accuracy of SSAE in comparison to the three existing methods was evaluated. The experiment was achieved using a small dataset consisting of two phones with flat images for training and natural images for testing. We refer to this experiment as case 5. Table 4.12 shows the lists, number of images and their original resolutions. All the images were center-cropped into 128 × 128 and 256 × 256 image sizes. Apart from the batch size, the same hyper-parameters used in all the SSAE experiments are also used in this experiment. Since the training size is 1200 compared to 3960 used in Table 1, we used a batch size of 8 for all image sizes.

Table 4.13 shows the comparative study of the identification accuracies of our proposed SSAE and the other three state of the art methods. Optimal overall identification accuracies were obtained for all the images sizes at 5th epoch. For 128 × 128, average identification accuracies of the two cameras are 95.30%, 95.30%, 95.30% and 97.70% for MLE SPN, Li's Model 5, Phase SPN and proposed SSAE respectively. For 256 × 256, average identification accuracies of the two cameras are 96.16%, 95.73%, 95.73% and 99.05% for MLE SPN, Li's Model 5, Phase SPN and the proposed method respectively. Our SSAE achieved higher average identification accuracy than the three existing methods using 128 × 128 and 256 × 256 image sizes for all the methods. Therefore, using PRNU of stronger quality can further increase the generalization capability of the proposed SAE with improved accuracy compared with these existing methods as shown for 128 ×128 and 256 × 256 image sizes.

Table 4.12. Details about the cameras used in the experiment including the resolution and number of images taken by each camera.

Camera Brand	Number of Flat Images	Number of Natural Images	Resolution
Samsung S7	600	117	2988 × 5312
Redmi Note 3	600	108	4608 × 3456

Table 4.13. Comparative study of the identification accuracies (in percentage %) of our proposed method and three other state-of-the-art methods (case 5).

Methods	128 × 128		256 × 256	
	Samsung S7	Redmi Note 3	Samsung S7	Redmi Note 3
MLE SPN	100.00	90.60	100.00	92.31
Model5	100.00	90.60	100.00	91.45
Phase SPN	100.00	90.60	100.00	91.45
Proposed method	100.00	**95.40**	100.00	**98.10**

4.4 Summary

Source camera identification based on stacked autoencoder for instance-based identification has been implemented in this research. The contribution of this novel approach to the body of knowledge is summarized below:

(i) The possibility of using deep learning module based on SSAE to solve source camera identification problem. (ii) Good generalization performance without the use of a huge dataset usually used in many deep learning implementations is achievable. Therefore, the proposed method is computationally effective. (iii) Based on the observation of the experimental results, reliable results were achieved especially when flat images were used for the training the network.

The intuition behind the proposed autoencoders is that, autoencoders are dataset specific and hence the trained features are learned using PRNU because of its uniqueness to each camera device. The proposed method achieves significant overall identification performance comparable with some existing methods on the Dresden database and better performance on our own dataset when compared with some state-of-the-art methods. Furthermore, the proposed network also generalizes well using the same hyper-parameters on different cameras' sets.

Chapter 5 :

Convolutional Neural Network as Feature Extractor and Classification for Source Camera Identification of Small Size Images

There are several CNN architectures that have been used in the literature [66, 69, 109-111]. They often consist of operations such as convolution, activation function, max-pooling and fully connected layers. In some applications, a very large number of convolutional layers were used to achieve good results. This increased the computation demand during training. Different CNN architectures in literature usually include different operations but the commonly used as shown in Figure 5.1. Several other signal processing operations can be used to improve generalization such as the use of stride or suitable normalization techniques. The suitability of some of these operations is dependent on the domain of adaptation or area of application. An operation such as max-pooling was replaced with the used of stride in [112]. The work in [112] used a stack of convolutional layers with a stride of two for image classification. Their proposed CNN architecture did not include the use of max-pooling nor fully connected layers. Despite that, their experimental results show better performance compared with other complex CNN architectures on the same dataset. In this work, we proposed deep convolutional neural network for instance-based source camera identification for classification and also as a feature extractor for one-vs-rest linear classifier. As PRNU fingerprint is positional dependent, it is important that the structure of CNN keeps information at each spatial location of cropped images of the same camera to be the same so as to achieve instance-based camera identification. Therefore, we extracted PRNU of single patch cropped from the center of the images as the input to the proposed network. After identifying suitable structures of CNN for the problem of instance-based camera identification, we compared our results with some state-of-the-art methods for source identification for small image size such as 64×64.

The rest of Chapter 5 is organized as follows. Section 5.1 discusses the working principle of CNN. Section 5.2 describes the training process for CNN. Section 5.3 describes the architecture of the proposed deep CNN for SCI and this includes, noise residues formulation for CNN, network architecture, the training procedures and the selection of fine-tuning parameters. Section 5.4 presents our experimental evaluation on Dresden database, which includes, the experimental settings, results and discussion and comparison with some existing state-of-the-art methods. Section 5.5 introduces our proposed fine-tuned pre-trained CNN for SCI, which includes, the idea of transfer learning, proposed methodology, experiment and results, comparison with some state-of-the-art methods. Section 5.6 compares our proposed CNN-based methods with a deep learning based method. Section 5.7 evaluates the robustness of the proposed deep CNN methods to post JPEG compression. Section 5.8 summarises the work and its contributions.

5.1 Working Principle of Convolutional Neural Networks

CNN shares a similar operation as traditional neural networks. They both have neurons with trainable weights. Because of the underlying assumption that the input data are images, some functions can be incorporated into the architectures. This, in turn, reduces the number of parameters in each layer [113]. In basic CNN structure, there are input layer, convolutional layers and the classification layers as shown in Figure 5.1. The input data fed into the CNN is a 3D image which consists of width, height, and depth. The width and the height correspond to the dimension of the image while the depth specifies the number of the channels of the image. For color images, the depth is 3 while it is 1 for gray-level images. The CNN architecture generates robust features of images and hence increases the capability of generalization. The convolutional layer mostly consists of three components: convolution, activations and pooling. The input image is sub-divided into different regions and an impulse signal is applied to each input region to form outputs. The impulse signal used is a filter and hence it is called a convolutional kernel or a filter kernel. The output shows the response of the filter in different spatial positions as the filters are applied locally in different image regions. The output of convolutional layers is often called the feature map. The convolution operation on the input and the filter kernel is expressed in [38] as,

$$o^{l}{}_{j} = \sum_{i=1}^{M} o_i^{l-1} \odot w_{ij}^{l-1} + b_j^l \tag{5.1}$$

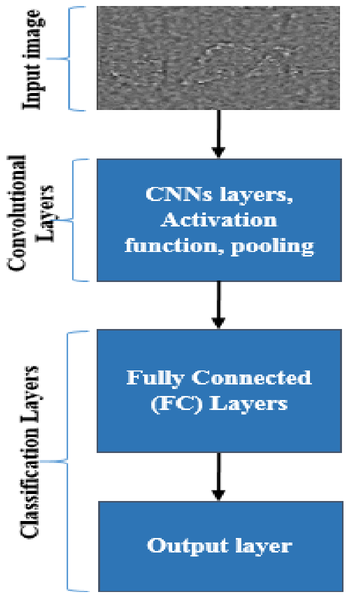

Input image
Convolutional Layers
CNNs layers, Activation function, pooling
Classification Layers
Fully Connected (FC) Layers
Output layer

Figure 5.1. Stages involved in conventional convolutional neural networks.

where $o^l{}_i$ is the j-th output in the l-th layer, w_{ij}^{l-1} is the weights of the filter kernel that connects the i-th node at the layer $l-1$ and the j-th node at the next layer l, $\odot$ is the convolution operation, b_j^l is the network bias of the j-th output neuron in the l-th layer, $i = 1, \dots M$ are the indices of the feature in each convolutional layer and M is the total number of the feature maps in the previous convolution layer. In [113], one of the important settings in the convolutional layer is the "stride". It is the number of times the filter kernels are shifted between pixels. A stride of one means that the filters are shifted by one pixel at a time which will lead to large output volumes. When the stride is two, filters are shifted by two pixels which will lead to double down-sampling. The next operation in the convolutional layer is the activation function. The aim of the activation function is to introduce non-linearity or randomness into the convolved output. The commonly used activation function for the CNN is the rectilinear unit (ReLU) [114, 115]. ReLU has been defined in eqn. (3.4). ReLU has been used in different CNN architectures for image classifications [66, 109, 116, 117]. Its advantage over other non-linear activation functions such as sigmoid and tanh is that it diminishes the tendency of having vanishing gradient and also increases the speed at which global convergence is attained [66]. Its ability to yield better performance is associated with the application of sparsity constraint on the output of a layer [84, 115]. Other variants of ReLU which have better performances than ReLU have been proposed and evaluated in [118]. These include the Leaky ReLU [119] and the parameterized ReLU (PReLU) [120]. Input data after normalization or other pre-processing techniques usually have both positive and negative samples. In PReLU, the gradient of the negative samples is also computed during optimization.

It is defined as,

$$PReLU(x) = \begin{cases} x & if\ x \geq 0 \\ ux & if\ x < 0 \end{cases} \tag{5.2}$$

where u is the weight associated with the input x. Instead of thresholding the input value to zero when it is smaller than zero, a weight (k) is associated with the input unit in the Leaky ReLU. Mathematically, it is defined as,

$$\text{Leaky } ReLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ kx & \text{otherwise} \end{cases} \tag{5.3}$$

The main difference between PReLU in eqn. (5.2) and Leaky ReLU in eqn.(5.3) is that u is learnt by back propagation while k is manually fixed or searched within a range of parameters. As reported in [118], the leaky ReLU out-performs both ReLU and PReLU activation functions when implemented on the CIFAR-10/CIFAR-100 dataset. The output volume after convolution operation increases with the number of filters used. However, the activation function does not affect the size of the output volume. The increase in output volume due to convolution usually results in a large number of network parameters. This can cause overfitting problem, i.e., the network may be biased when it generalizes on an unseen data. To reduce the number of network parameters, pooling operation is carried out on the output image. In fact, the pooling operation is a downsampling operation. The commonly used pooling techniques are the mean or maximum-pooling (max-pooling). To ensure that the feature maps and the features of the testing images have the same feature scaling, the feature maps are usually normalized. The last stage of the CNN is the classification stage which consists of fully connected layers. It is called fully connected layers because each neuron in the layer is fully connected to all the neurons in the previous layers. The last layer in the fully connected layer is an output layer. The output layer uses the softmax classifier and it is sometimes referred to as the softmax layer. Besides, multi-perceptron can be used in the fully connected layers. The number of neurons in the output layer is equal to the number of classes or targets. For source camera identification, the number of neurons in the softmax layer equals to the number of source cameras to be identified.

5.2. Training Process of CNNs

In this section, the training process of CNNs is discussed. We will describe how the CNN layers learn its weights and map the activated features. The training is supervised in which CNNs are trained and fine-tuned with training data and their corresponding labels. The training is done using back error propagation [102]. First, initial weights of the convolutional layers are randomly generated. Second, there are four steps in the training, namely the forward propagation, objective function or cost function calculation, backward pass and the network weight update. During forward propagation, the input image goes through both the convolutional layers and the classification layers. The output of the softmax classification during the forward propagation may have a large deviation from the expected output. This is because the weights of the network are randomly assigned. The objective function finds the deviation between the input features and the reconstructed features. Common cost functions include the sum of squared errors, cross-entropy (CE) and the exponential cost functions [86, 87]. The CE loss is usually adopted when softmax classifier is used for supervised fine-tuning because it avoids the changes in network weights to be decreasing to zero. Afterward, a backward pass is carried out from the output layer back to the input layer. Its aim is to determine the error of the weights in each layer of the network where the estimated error is the biggest. To correct this, gradients of the weights in each layer are estimated and all the weights of the networks are updated to reduce the estimated error. There are different gradient optimization algorithms adopted for training. For example, the batch gradient descent, stochastic gradient descent, mini-batch gradient descent, momentum, Nesterov accelerated gradient, Adagrad, Adadelta, RMSprop and Adam [96]. Training is repeated over several numbers of epochs until the training error is minimized over the training dataset.

As earlier explained, CNN architecture involves a lot of network parameters. A large number of parameters will require a large number of training samples and may have an over-fitting problem. A newly developed technique for preventing overfitting is the dropout regularization [98]. It is achieved by randomly setting a fraction of the units of a layer to zero while the remaining units in the layer will be used as the input to the next layer. The basic idea of the dropout regularization is to prevent the network from learning redundant features at each hidden layers so that only useful features are retained for generalization [93]. It also helps to reduce computational complexity by only performing activation on useful features.

5.3 Proposed CNN Architecture for Source Camera Identification

In this work, CNN is proposed to solve the instance-based source camera identification problem. The proposed CNN-based system retains the positional correspondence information about the sensor noise pattern for reliable device identification. Besides, we focus on identification of small image size as existing approaches cannot provide reliable identification in such cases. In this section, details about the proposed CNN-based source identification system will be discussed.

This includes the extraction of the noise residues of the image, the architecture of the proposed system and the fine tuning steps.

5.3.1 Noise Residues Extraction and Formulation for CNN

PRNU is proposed as the input feature as earlier explained. The general pixel output model based on the sensor's imperfections is given in eqn. (2.2). The resulting noise residue W from eqn. (2.2) is also expressed in eqn. (2.3). The denoising filter proposed by Lukas et al. [5] was used to denoise the cameras' images. To obtain a reliable estimate of the PRNU image, state-of-the-art algorithms rely on obtaining a set of photos taken by the camera. From this set of photos, PRNU image is obtained either by averaging their noise residues or through the maximum-likelihood estimation. In our proposed CNN-based source identification method, the noise residues are the inputs to the deep learning network so that features inside the noise residues are learned by the network.

Different pre-processing techniques are usually carried out on the data. The noise residues are pre-processed by zero mean processing. It means that each row of the noise residue is first subtracted from its row average, and then each column is subtracted from its column average to give the normalized noise residue. Given that the size of the normalized noise residue is $\widetilde{w}$. All the noise residues of images belonging to the ith camera will be concatenated and expressed as,

$$T_i = [\widetilde{w}_1, \widetilde{w}_2, \ldots, \widetilde{w}_{n_i}] \tag{5.4}$$

The size of T_i is $d^2 \times n_i$,where n_i is the total number of noise residues for the ith camera. Given that M is the total number of cameras to be identified, the total number of samples for all the cameras used in the experiment (T_S) can be expressed as,

$$T_S = [T_1, T_2, T_3 \ldots \ldots . T_M] \tag{5.5}$$

When the number of images is the same for all cameras, i.e.,$n_1 = n_2 = n_M$, this becomes a balance classification problem. Otherwise, it is an imbalanced classification problem. The camera labels are formed by using one-hot encoding. This means that the column vector representing a noise residue consists of only one non-zero element whose position indicates the camera that was used to produce that image.

5.3.2 Network Architecture of Our Proposed CNN for Source Camera Identification

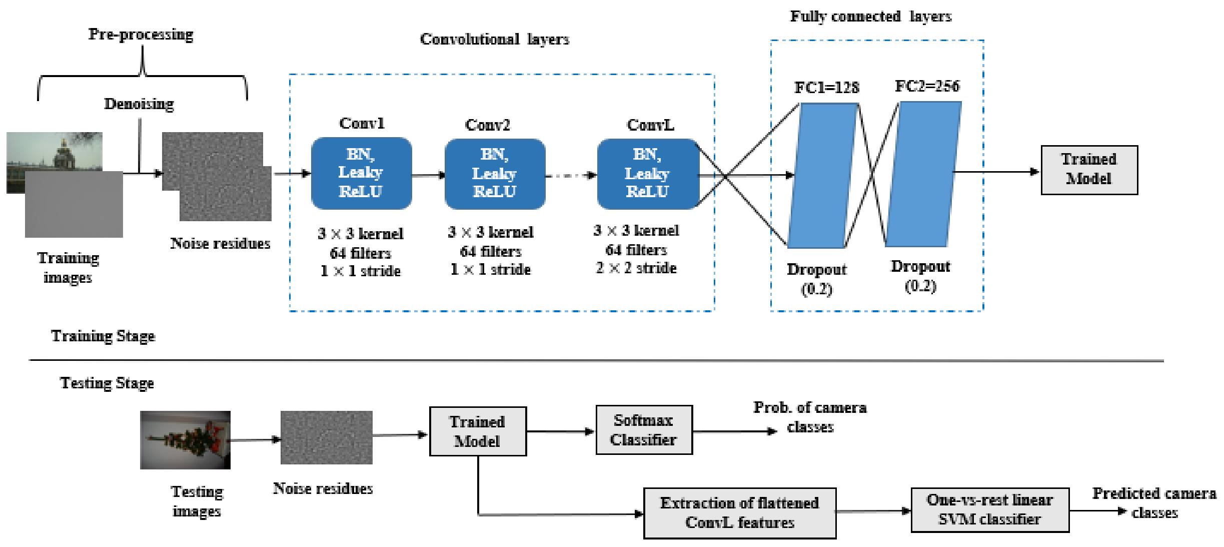


Figure 5.2. The layout of the CNN for instance-based camera source identification.

Note: The flattened output of the optimal convolutional layer is extracted as given as an input to a one-vs-rest SVM classifier.

Figure 5.2 shows the proposed CNN architecture for instance-based SCI. It consists of the following components: L number of convolutional layers, fully connected layers, softmax and one vs-rest linear SVM classifiers. Each convolutional layer consists of convolution operation, activation function and batch normalization. The noise residues are first extracted from images using eqn. (5.5). As explained in Section 5.3.3, the normalized noise residues are used as input into the first convolutional layer, Conv1. The output of the Conv1 is then given as input to the next convolutional layer, Conv2. This process is repeated until the L-th convolutional layer, ConvL, to provide a good feature representation. In all the convolutional layers, Leaky ReLU is used as the activation function due to its fast computation and superior performance as compared to ReLU and PReLU [118]. Moreso, ReLU considers only positive samples of convoluted features while Leaky ReLU considers the signal information of both positive and negative values. PRNU signal after the zero-mean operation consists of both negative and positive values and hence, using ReLU will remove some of the PRNU signal information. The weight is set to 0.001 as in [119] for Leaky ReLU. In order to increase the rate of convergence during training, batch normalization (BN) is used as a pre-processing technique on the convoluted output. It is called batch normalization since it is being carried out on small batches of the convoluted output. The variation between the trained features and test features are greatly reduced and it helps to increase the overall network generalization accuracy. This was applied prior to the activation function in each convolutional layer. There is always a decrease in the rate of convergence during training due to the internal co-variance difference in the distribution of each hidden layer. In order to increase the rate of convergence during training, batch normalization (BN) was used as a pre-processing technique on the convoluted output. It is called batch normalization since it is being carried out on small batches (N_B) of the convoluted output [121]. Given that, $N_B = (x_1,\ x_{2,}\, x_3 \ldots .\, x_B)$, the normalization of each dimension $(\hat{x})$ can be expressed as,

$$\hat{x} = \frac{x_b - E(x_b)}{\sqrt{\sigma_x}} \tag{5.6}$$

where x is the input of a hidden layer,$b = 1, 2,\ldots B$, σ_x and $E(x_b)$ are the estimated variance and expectation over a mini-batch over a training set. The scaled and shifted normalization value can be expressed as,

$$y = \zeta\hat{x} + \delta \tag{5.7}$$

where, ζ and δ are learnable parameters. Batch normalization generates activations that have Gaussian distribution and this further help emphasizes the Gaussian distribution nature of the PRNU fingerprints of cameras.

In our work, the max-pooling operation was replaced with the use of stride since max-pooling is mostly used in applications where translation-invariance is desired. The use of PRNU as the input to the CNN does not require accounting for translational invariance since only the center parts of the camera images were used. Furthermore, max-pooling aggressively down-sample features and the quality of the feature maps since PRNU signal is a pixel-strength dependent signal. The stride of 1 was used in the first two convolutional layers of our network so as to keep the spatial size of images to prevent loss of information. To reduce the number of parameters, we used the stride of 2 in Conv3 which acts as down-sampling operation. The output volume of the convolutional layer depends on the number of filters kernels used. Given that the size of the input image is W $\times$ H, the output of the convolutional layer will be $W \times H \times n_{b\_}filter,$ where $n_{b\_}filter$ is the number of filter kernels used. The choice of $n_{b\_}filter$ depends on the size of the input data. Some of the optimal parameters are obtained during experiments and their effectiveness for network generalization can be carried out using cross-validation [122]. Before the fully connected layers, flattening is carried out on the output of the last convolutional layer. Flattening refers to the output of the convolutional layer that is converted to a one-dimensional vector.

Firstly, the flattened output is given as input to two fully connected layers; namely FC1 and FC2. There is no standard method for deciding the number of neurons in the fully connected layers. This can be empirically determined during experiments. However, the numbers are dependent on the size of the input data. The neurons in FC2 are fully connected to a softmax classifier for probabilistic predictions of camera classes. The number of neurons in the softmax classifier is equal to the number of cameras used.

Secondly, as shown in Figure 5.2, after the training of the proposed network with FC1, FC2 and the softmax classifier, we neglected these layers and extracted the output of the Conv3 after flattening with a linear output and used this as the embedded layer for a one-vs-rest linear SVM classifier [123] for the prediction of camera classes. One-vs-rest linear SVM classifier is also known as the one-vs-all SVM classifier. SVM score is obtained by computing the probability that a given data point belongs to a particular class by Platt scaling [124]. Plat scaling is used in transforming the outputs of a classification model into a probability distribution over classes. For one-vs-rest, each linear classifier will be trained by the entire camera classes. All the samples of a one specific camera class will be treated as class one (C_1) and the all other samples of the remaining camera classes are treated to belong to a single class (C_r). Unlike one –vs-one classifier that uses $C(C-1)/2$ linear classifiers, one-vs-rest uses only K linear SVM classifiers. During testing, the class with maximum score among the C classifiers is the class belonging to the testing sample. Where C is the total number of classes. Using one-vs-rest linear SVM classifier gives room for more training samples in a training set for each phase of training. Though, this increases computation time during training especially when the number of target cameras is large [125].

Since the number of the nodes after flattening is larger than the number of training samples in our experiments, we are motivated to use one-vs-rest linear SVM classifier. Also, one-vs-rest linear SVM is less sensitive to datasets with unequal samples.

Though, One-vs-rest linear SVM is more computationally expensive compared to using softmax classifier for prediction of classes. We implemented the one-vs-rest linear SVM using scikit learn library in Keras.

5.3.3 Fine-Tuning and Training of the Proposed Network

The fine-tuning process can be described as an iterative method of finding filter weights (w) that can help in the minimization for the CNN's cost or objective function. Given x as the training data, N as the number of training data, the error rate $(\mathrm{J}(\mathrm{W},\mathrm{x})$ is expressed in [122] as,

$$\mathrm{J}(\mathrm{W},\mathrm{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{L}\,(f(\mathrm{x}_i, w), c_i) + \qquad (5.8)$$

where, L is the loss function, x_i is the ith image of x, $f(\mathrm{x}_i, w)$ is the function for predicting the class c_i of x_i given w. In all our experiments, the mini-batch stochastic gradient descent was used to determine the optimal weights of the CNN and L was used as the categorical loss function. In each iteration, the optimal weight was updated. Given that the current weight is w_k, the updated weight in each iteration is expressed in [122] as,

$$\mathrm{w}_{k+1} = \mathrm{w}_k + \eta\left[\gamma \Delta \mathrm{w}_k - \frac{\partial \mathrm{J}(\mathrm{W},\ N_B)}{\partial k} - \lambda \mathrm{w}_k\right] \tag{5.9}$$

where η, γ, λ , N_B are the learning rate, momentum coefficient, weight decay and the mini-batch size respectively. The function of the momentum is to control the learning rate so that a fast global convergence can be achieved. A decay rate was also used together with the momentum. The decay rate is used for the modification of the learning rate (η) to further reduce the error rate. The decay rate may be obtained by dividing the learning rate with the number of epochs used. The term epoch means a single training pass (weight update) using all the training set [122]. Since, the proposed Dresden dataset used in our experiments is an unbalanced dataset, we introduced the use of class weight to the training function. Class weights penalize under or over-represented classes in the training set. The class weight was calculated by using $class\ weight =\ log\ (total\ samples/ samples\ in\ a\ class)$. The log function helps to smoothen the weights for every imbalanced class. If the class weight is less than 1, estimated weight for the class will be used as 1. The class weights can be fine-tuned by a smaller parameter, p. We used p as 0.15. This is then passed as a dictionary into the class weight parameter of the network training function. Furthermore, in order to prevent overfitting of network and reduce unnecessary computation during training, we adopted the use of early stopping [126]. In our experiments, since we are interested in identification accuracy of classes, we monitored the training accuracy and initiated the training to stop once there is no change in the training accuracy over two consecutive epochs. In order to ensure that the parameters used in our experiments do not have high variance when evaluated on another set of testing data, we used K-fold cross-validation and computed the mean accuracy over the entire testing sets. K was set as 10 in all experiments. To prevent overfitting, we further used dropout regularization, weight and sparsity regularizations. The weight and sparsity regularizations were added to FC1, FC2 and the softmax layer. Dropout regularization was used with the fully connected layers (FC1 and FC2) only.

5.3.4 Selection of Fine-Tuning Parameters

Selection of fine-tuning the parameters is mostly done by empirical methods. Most commonly used methods are grid search, hold-out validation datasets, random search and the use of proven parameters in existing literature. In our work, the knowledge of existing parameters and searching of the optimal parameters within ranges of parameters were adopted. The optimal performance for λ is set at 10^{-5}. Parameters to be fine-tuned are, γ, λ , N_B and the number of epochs. η is searched between 0.01 and 0.005. For all the experiments, irrespective of the input size, the optimal results are usually obtained when η is set as 0.001. γ is searched between 0.5 and 0.9. The optimal performance in all experiments is at 0.9 irrespective of the image sizes. N_B of a power of 2, i.e., 16, 32, 64, 128 and 256 are usually used to ensure that the GPU memory has the greatest usage [122]. Due to the image sizes used in our experiments and GPU memory constraints, smaller N_B was tested unlike 256 commonly used in literature [122, 127]. One of the easiest ways of reducing memory consumption is using reduced batch size. We obtained the optimal results in our experiments using 16 for 64×64 image size. The time taken to complete a mini-batch is one iteration. Both the weight and sparsity regularization parameters are searched at range of 10^{-6} to 10^{-2}.

5. 4 Experiments and Results

5.4.1 Experimental Settings

The same Dresden database and 20 cameras used in Table 4.1 are also used for the proposed methods. The only difference is that, we also added the flat images of each camera as part of our experimental evaluation. However, not all the used cameras in the Dresden database have flat images. Table 5.1 shows the cameras with their natural and flat images. The total number of flat and natural images are 4996 and 750 respectively. Flat images can only be used for training and not used as test data for source camera identification problem. Therefore, since we used cross-validation during training, noise residues of the flat image with corresponding labels were only added to each training fold of the natural images. Apart from flat images having quality PRNU fingerprints, it also helps to increase the size of the data and this is an added advantage when training a deep network. Since, conventional methods have lower camera identification accuracy for small sizes images, the center part of the images was cropped to produce 64×64 image sizes. All experiments are carried out on the same experimental platforms discussed under Section 4 3.1.

5.4.2 Results and Discussion

The proposed CNN-based source camera identification system shown in Figure 5.2 has a few components. Table 5.2 shows the list of components of the proposed CNN architecture used for source camera identification for 64×64 image size along with its main parameters. In all our experiments, optimal performance was obtained using three convolutional layers at a batch size of 16. The epoch was set to 20 and as earlier stated and early stopping with a patience of 2 was used in all experiments. To investigate how the proposed method generalizes well when new cameras are used and to show that the parameters are not dataset specific, we divided the cameras set into three cases. In case 1, the experiment was carried out on the first ten cameras in Table 5.1, for case 2, the experiment was carried out on the remaining ten cameras, i.e., cameras 11 to 20 in Table 5.1 and in case 3, the experiment was carried out on all the 20 cameras. The significance of the division of the cameras to cases 1 and 2 is to be able to see how the hypermeters used in case 1 generalizes when used on new sets of cameras in case 2. Finally, the case 3 shows the effects of an increase in number of cameras on the identification accuracies of cameras. Experiments were carried out on each case. Experiments in each case include the use of CNN both for classification and also as a feature extractor for one-vs-rest linear SVMs. In the remaining part of this work, the CNN with softmax classifier and also as feature extractor with one-vs-rest linear SVM are named as CNN-SC and CNN-SVM respectively. The results of the three cases are grouped and explained under the three sub-sections of experiments below.

Table 5.1. Details about the cameras used in the experiment including the resolution and number of flat and natural images.

S/N	Camera Brand	Resolution	Natural Images	Flat Images
1	Agfa_DC-830i	3264 × 2448	342	-
2	Olympus_mju_1050SW_0	3264 × 2736	204	50
3	Olympus_mju_1050SW_1	3264 × 2736	209	50
4	Kodak_M1063_0	3664 × 2748	464	-
5	Kodak_M1063_1	3664 × 2748	458	-
6	Agfa_Sensor530s_0	4032 × 3024	372	-
7	Nikon_D70_0	3008 × 2000	180	25
8	Nikon_D70_1	3008 × 2000	189	25
9	Olympus_mju_1050SW_2	3264 × 2736	218	50
10	Canon Ixuss 55	2592 × 1944	224	50
11	Canon_Ixus70_0	2304 × 3072	171	50
12	Canon_Ixus70_1	2304 × 3072	179	50
13	Canon_Ixus70_2	2304 × 3072	171	50
14	Samsung_L74wide_0	2304 × 3072	229	50
15	Samsung_L74wide_1	2304 × 3072	224	50
16	Samsung_L74wide_2	2304 × 3072	231	50
17	Samsung_NV15_0	2304 × 3072	217	50
18	Samsung_NV15_1	2304 × 3072	214	50
19	Sony_DSC-H50_0	2736 × 3648	266	50
20	Sony_DSC-H50_1	2736 × 3648	234	50
Total number of Images			**4996**	**750**

Table 5.2. Components of CNN architecture and its parameters.

S/N	**Layer Component**	**Component parameter**	**Value**
			64×64
1	Convolution	Kernel size No. of filters Stride size	$3 \times 3 \times 1$ 64 1×1
2	Batch Normalization	Axis	1
3	Leaky ReLU	weight (k)	0.01
4	Convolution	Kernel size No. of filters Stride size	$3 \times 3 \times 1$ 64 1×1
5	Batch Normalization	Axis	1
6	Leaky ReLU	weight (k)	0.01
7	Convolution	Kernel size No. of filters Stride size	$3 \times 3 \times 1$ 64 2×2
8	Batch Normalization	Axis	1
9	Leaky ReLU	weight (k)	0.01
10	Fully Connected	Units	128
		Dropout rate	0.2
		Reg. parameter	10^{-5}
11	Fully Connected	Units	128
		Dropout rate	0.2
		Reg. parameter	10^{-5}
12	Softmax Classifier	Units (No. of camera classes)	10 & 20
13	One-vs-rest linear SVM	Units (No. of camera classes) Random state	10 & 20 0

Experiments 1 (Case 1)

The overall accuracy of each split of testing data for both CNN-SC and CNN-SVM for case 1 are shown in Table 5.5. By overall accuracy we mean, the ratio of the number of correctly identified test samples of all cameras to the total number of test samples of all cameras in each fold. The standard deviations of accuracies over the 10 folds cross-validation for both CNN-SC and CNN_SVM are $\pm$ 2.28%, $\pm$ 0.02% respectively. The average overall accuracies are 67.48% and 69.69% for both CNN-SC and CNN-SVM respectively. These results show that the CNN-SVM has improved performance of 2.21% over the CNN-SC for case 1. Hence, the confusion matrix for the first ten cameras using CNN-SVM is shown in Table 5.3. The individual camera identification accuracy is shown by the diagonals of Table 5.3 in bold. The average identification accuracy for 10 cameras for an image size of 64×64 is 69.01%. The cameras devices; Olympus_mju_1050SW_1 and Nikon_D70 are observed to have lower identification accuracies as compared to other cameras. Nikon camera is always reported to be difficult, as it has diagonal artifacts which affect the accuracy. The generalisation performance of deep learning networks is also dependent on the quality of the input features.

Experiments 2 (Case 2)

For case 2, the parameters of the network are not reset. The same network and fine-tuning parameters used in case 1 are directly applied to the cameras in case 2. The overall accuracy of each split of testing data for both CNN-SC and CNN-SVM for case 2 are shown in Table 5.5. The standard deviation of accuracies over the 10 folds cross-validation for both CNN-SC and CNN-SVM are $\pm$ 3.64 % and $\pm$ 0.03% respectively. The average overall accuracies are 71.91% and 76.73% for both CNN_SC and CNN-SVM respectively. These results show that the CNN-SVM has improved performance of 4.82% over the CNN-SC for case 2. This means, for both cases 1 and 2, the proposed CNN-SVM has better generalisation accuracy compared to using CNN-SC. Hence, Table 5.4 shows the confusion matrix for cameras in case 2 using CNN-SVM. The identification accuracy of each camera is shown by the diagonals of Table 5 in bold. The average identification accuracy of the cameras in case 2 is 76.30%. The superior performance of Sony_DSC-H50_0 and Sony_DSC-H50_1 compared to other cameras can be attributed to the quality of the noise residues of the Sony camera brand. Experiments on case 2 show that the parameters used in case 1 are not data specific and can be applied directly to new sets of cameras. However, the average identification accuracy of case 2 for CNN-SVM is 6.61% greater than that of case 1. The higher accuracy obtained for case 2 was because many source cameras in case 2 have high identification accuracies. This may be due to the quality of natural images of those cameras as PRNU signal can best be extracted from high-intensity images with simple texture complexity.

Experiments 3 (Case 3)

Furthermore, we carried out experiments for all the 20 cameras in Table 4.1. The overall accuracy of each split of testing data for both CNN-SC and CNN-SVM for case 3 are shown in Table 5.5. The standard deviation and accuracies over the 10 folds cross-validation for both CNN-SC and CNN-SVM are $\pm$ 1.29 % and $\pm$ 0.02% respectively. The average overall accuracies are 66.93 % and 70.28% for both CNN-SC and CNN-SVM respectively. Since the CNN-SVM has the highest average overall accuracy, we itemized each camera identification accuracy of proposed network using CNN-SVM in Table 5.6. Table 5.6 also shows the number of testing images and correctly identified images for the 20 cameras. The average identification accuracy for the 20 cameras is

69.79%. The experimental results show that the average identification accuracy for case 3 is 0.78% greater than case 1 and 6.51 % lesser than case 2. In machine learning, the more the number of classes is, the lower the overall identification accuracy is. The higher accuracy obtained in case 3 compared to case 1 is due to the strong presence of noise residues of cameras of case 2 in case 3. Figure 5.3 shows the variation of the error rate with respect to the number of epochs over all the 10 folds. It was observed in Figure 5.3 that the error rate becomes stable before getting to 20 epochs in all the 10 folds of training data due to the early stopping applied during training. We also investigated the performance of our proposed methods for the three cases with and without the use of flat images as part of training data. Table 5.7 shows the overall accuracies of proposed methods with and without the use of flat images in the training process. As observed from Table 5.7, for CNN-SC, there is an increased overall identification accuracy of 4.76%, 6.65% and 3.00% in case 1, case 2 and case 3 respectively when the flat images were included as part of training data. While for CNN-SVM, there is an increased overall identification accuracy of 3.4%, 4.16% and 3.87% in case 1, case 2 and case 3 respectively when the flat images were included as part of training data. The increase in accuracy can be attributed to the quality of noise residues extracted when flat images are used and also, generalization accuracy increases with increased training data in machine learning problems. However, in a situation where there is no availability of flat images for cameras under investigation, the proposed methods still have good performance with the use of natural images as training data only.

Table 5.3. Identification accuracy (in percentage %) of the proposed method for 64 × 64 image size (case 1).

Camera Device		1	2	3	4	5	6	7	8	9	10
Agfa_DC-830i	1	**82.75**	0.29	-	1.46	1.17	6.43	2.63	5.26	-	-
Olympus_mju_1050SW_0	2	-	**63.24**	17.16	1.47	-	1.47	-	-	16.67	-
Olympus_mju_1050SW_1	3	1.91	18.18	**57.89**	1.91	0.48	0.48	-	0.48	18.66	-
Kodak_M1063_0	4	2.59	0.22	-	**60.56**	32.11	2.59	0.22	1.08	-	0.65
Kodak_M1063_1	5	3.49	-	-	32.31	**61.79**	0.66	0.22	0.87	-	0.66
Agfa_Sensor530s_0	6	1.88	-	-	1.61	0.81	**87.37**	5.11	3.23	-	-
Nikon_D70_0	7	3.33	-	0.56	2.22	1.67	16.67	**53.89**	20.00	1.11	0.56
Nikon_D70_1	8	1.59	-	-	3.70	0.53	15.87	17.99	**59.26**	-	1.06
Olympus_mju_1050SW_2	9	1.38	16.51	15.14	1.38	0.46	0.46	0.46	0.00	**64.22**	-
Canon Ixuss 55	10	0.45	-	-	-	-	0.45	-	-	-	**99.11**

Table 5.4. Identification accuracy (in percentage %) of the proposed method for 64 × 64 image size (case 2).

Camera Device		11	12	13	14	15	16	17	18	19	20
Canon_Ixus70_0	11	**74.27**	12.28	9.94	-	-	-	-	2.34	0.58	0.58
Canon_Ixus70_1	12	11.73	**76.54**	8.38	-	-	-	1.68	0.56	-	1.12
Canon_Ixus70_2	13	9.36	9.94	**73.10**	0.58	1.17	1.17	1.17	1.17	1.17	1.17
Samsung_L74wide_0	14	-	-	-	**70.74**	12.66	13.54	1.75	0.87	0.44	-
Samsung_L74wide_1	15	0.45	-	-	16.52	**64.29**	15.18	1.34	1.79	0.45	-
Samsung_L74wide_2	16	-	0.87	0.43	12.12	16.45	**67.10**	0.43	2.60	-	-
Samsung_NV15_0	17	-	-	0.46	1.38	2.76	1.38	**79.72**	11.98	0.46	1.84
Samsung_NV15_1	18	-	-	-	1.40	2.80	1.40	13.08	**76.64**	2.34	2.34
Sony_DSC-H50_0	19	-	0.38	0.38	0.38	-	-	0.38	0.75	**91.73**	6.02
Sony_DSC-H50_1	20	-	0.43	0.85	0.43	0.85	-	0.85	1.28	6.41	**88.89**

Table 5.5. The overall accuracy of each testing fold for both CNN-SC and CNN-SVM For Case 1, Case 2 and Case 3.

Fold No.	**Case1**		**Case2**		**Case3**	
	CNN_SC	**CNN_SVM**	**CNN_SC**	**CNN_SVM**	**CNN_SC**	**CNN_SVM**
1	67.5	67.1	69.6	75.7	68.0	71.4
2	70.6	67.5	74.8	77.1	67.0	67.6
3	65.7	70.3	75.7	78.5	65.8	72.4
4	64.3	68.9	71.0	73.8	64.4	72.2
5	68.9	71.0	72.0	80.8	68.4	71.2
6	65.0	66.4	70.1	77.6	66.6	72.2
7	68.2	68.9	64.8	71.8	67.9	69.1
8	70.6	72.0	73.2	76.1	65.5	67.3
9	69.2	70.3	78.4	80.3	68.5	68.1
10	64.7	74.5	69.5	75.6	67.1	71.1
Std. dev. (± %)	**2.28**	**0.02**	**3.64**	**0.03**	**1.20**	**0.02**
Average accuracy (%)	**67.48**	**69.69**	**71.91**	**76.73**	**66.93**	**70.28**

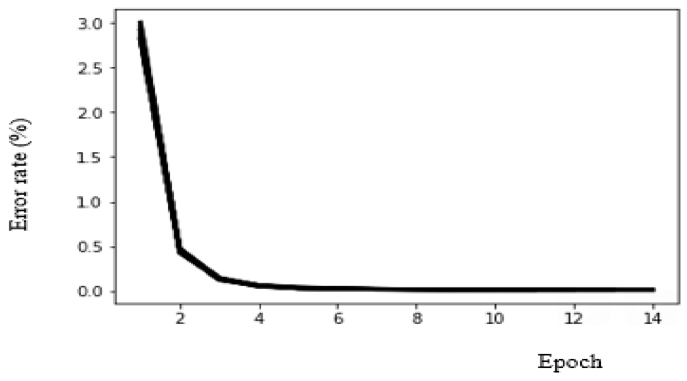


Figure 5.3. Error rates against the number of epochs for the 20 cameras.

Table 5.6. An overall number of testing images, correctly identified images, and the overall identification accuracy for CNN-SVM for the 20 cameras.

S/N	**Camera Device**	**Testing images**	**No. of correctly identified testing images**	**Identification Accuracy**
1	Agfa_DC-830i	342	275	80.4
2	Olympus_mju_1050SW_0	204	126	61.8
3	Olympus_mju_1050SW_1	209	117	56
4	Kodak_M1063_0	464	273	58.8
5	Kodak_M1063_1	458	290	63.3
6	Agfa_Sensor530s_0	372	320	86
7	Nikon_D70_0	180	102	56.7
8	Nikon_D70_1	189	103	54.5
9	Olympus_mju_1050SW_2	218	144	66.1
10	Canon Ixuss 55	225	191	85
11	Canon_Ixus70_0	171	117	68.4
12	Canon_Ixus70_1	179	131	73.2
13	Canon_Ixus70_2	171	107	62.6
14	Samsung_L74wide_0	229	169	73.8
15	Samsung_L74wide_1	224	143	63.8
16	Samsung_L74wide_2	231	162	70.1
17	Samsung_NV15_0	217	149	68.7
18	Samsung_NV15_1	214	149	69.6
19	Sony_DSC-H50_0	266	239	89.8
20	Sony_DSC-H50_1	234	204	87.2
Total no:		**4997**	**3511**	**1395.8**
Overall Accuracy (%)			**70.26**	**69.79**

Table 5.7. Overall accuracies of proposed methods with and without the use of flat images in the training process (%)

Proposed method with and without flat images	**Case 1**	**Case 2**	**Case 3**
CNN-SC without flat images	64.93	65.26	63.69
CNN-SC with flat images	67.48	71.91	66.93
CNN-SVM without flat images	66.29	72.57	66.41
CNN-SVM with flat images	69.69	76.73	70.28

5.4.3 Comparison with some state-of-the-art methods

In this part, we compare our proposed method (CNN-SVM) with four state-of-the-art methods for instance-based SCI for both 64×64 size. The methods are the maximum likelihood estimated SPN (MLE SPN) [8], Li's model SPN [9], Phase SPN [10] and weighted averaging (WA) method[51]. Model 5 of the six proposed Li's models gives the best overall accuracy and hence it is used in our comparative study. The weighting average using MLE (MLE WA) proposed in [51] was used. For the purpose of fair comparison among the four methods, the identification was achieved using the peak to energy correlation [10]. Since we are not using cross-validation for the state-of-the-art methods like we did for CNN-SVM and for the purpose of fair comparison, the flat images were included with the 80% of the natural images for the training of the proposed CNN-SVM. The remaining 20% of the natural images were used for testing. The same experimental setting was used for the compared state-of-the-arts methods. The comparison results are shown in Table 5.8. It was shown that the average accuracy per camera device of the proposed method is 23.03%, 25.26%, 18%, and 21.61% higher than that of MLE SPN, LI’s model 5, Phase SPN and WA method respectively.

Table 5.8. Comparative study of the identification accuracies (in percentage %) of our proposed method (CNNN-SVM) and four other state-of-the-art methods for 64 × 64 image sizes.

S/N	**Camera Device**	**MLE**	**Li's Model 5**	**Phase SPN**	**WA**	**CNN-SVM**
1	Agfa_DC-830i	63.77	59.27	60.87	63.77	**88.14**
2	Olympus_mju_1050SW_0	31.70	39.02	36.59	34.15	**73.17**
3	Olympus_mju_1050SW_1	50.00	52.38	**57.14**	45.24	45.83
4	Kodak_M1063_0	22.58	15.05	16.13	30.12	**62.22**
5	Kodak_M1063_1	31.52	25.00	27.13	33.70	**66.29**
6	Agfa_Sensor530s_0	69.33	66.67	64.00	81.33	**84.34**
7	Nikon_D70_0	30.56	27.78	22.22	19.44	**65.85**
8	Nikon_D70_1	34.21	31.58	26.32	21.05	**54.05**
9	Olympus_mju_1050SW_2	50.00	63.63	56.82	47.72	**66.67**
10	Canon Ixuss 55	33.33	24.44	46.67	**86.67**	81.81
11	Canon_Ixus70_0	71.43	65.71	**74.29**	25.71	62.50
12	Canon_Ixus70_1	55.56	55.56	55.56	44.44	**65.85**
13	Canon_Ixus70_2	60.00	68.57	**77.14**	60.00	65.63
14	Samsung_L74wide_0	30.43	23.91	56.52	43.48	**72.55**
15	Samsung_L74wide_1	13.33	17.78	31.11	31.11	**71.12**
16	Samsung_L74wide_2	23.40	12.77	55.32	31.91	**67.64**
17	Samsung_NV15_0	47.73	31.82	47.73	36.36	**66.00**
18	Samsung_NV15_1	66.79	55.81	62.79	**76.74**	75.76
19	Sony_DSC-H50_0	79.63	79.63	85.19	79.63	**86.54**
20	Sony_DSC-H50_1	78.72	82.98	**85.11**	80.85	82.69
Average accuracy per camera device		47.20	44.97	52.23	48.67	**70.23**

An improved identification accuracy is observed for 14 out of the 20 camera devices using CNN-SVM compared to other methods. This shows that our proposed CNN architecture can extract robust sensor characteristics from small image size. As earlier mentioned under motivation for our work in Section 1.2 of Chapter 1, the CNN-based method [19] published in 2019 used PRNU images of cameras as input data and has lesser identification accuracy than the compared PRNU based method [5]. The work in [19] used max-pooling in all the three convolutional layers and also spatial pooling for dimensionality reduction of feature maps. These operations especially max-pooling does aggressive down-sampling, and this greatly affects the strength and pixel-positional correspondence nature of the individual PRNU images of cameras. Hence, our architecture excluded these pooling operations and only used strided convolution for better network generalization. Our experimental results in Table 5.8 shows that our proposed well-designed CNN-based methods with effective training algorithms have better identification accuracies than the four compared PRNU-based methods using the same number of training and testing images. Hence, our proposed CNN-based methods have better generalization capability than the CNN-based method proposed in [19] and hence can achieve higher identification accuracy than PRNU-based techniques for SCI.

5.5 Fine-tuned Pre-trained Convolutional Neural Networks for Source Camera Identification

In this section, we aim to fine-tune already pre-trained proposed CNN architecture for source camera identification in Figure 5.2. In the remaining part of this section, we will give brief background understanding on transfer learning through fine-tuning of the pre-trained network, discuss the modification to the proposed method in Figure 5.2 and finally, give experimental results and discussion.

5.5.1 The Idea of Transfer Learning and Our Motivation

In recent times, it is rare to train CNN from the scratch because of the need to use large dataset for training so as to prevent overfitting. The usual approach adopted to solve the problem of overfitting is fine-tuning a model trained on a large dataset on a different dataset through the use of backpropagation. In transfer learning, the problem over which the pre-trained network was trained is called the source problem and the problem at hand is the target problem. If there will be a better generalization of the target problem, the source problem and the target problem must have similar or related distribution. This approach of using similar or related distribution is called domain adaptation. There are still challenges in obtaining good generalization accuracy for a target problem if it has different distribution compared to the source problem [102]. Most of the available pre-trained networks are networks trained on image classification problems. Examples of such pre-trained network are Alex[66] and GoogleNet [109]. Those pre-trained networks extracted their features directly from the images. For SCI, our aim is not to identify the images but the source cameras of the images and hence, we need to learn specific features related to each camera. Also, most of the images in the Dresden database were captured from the similar scenes using different cameras and hence learning just the image features will only mean we are learning correlated features. This was why we earlier proposed PRNU fingerprints as the input data for our proposed CNN architecture in Figure 5.2. This is because PRNU fingerprints are distinct to each camera. Therefore, if CNN model pre-trained on PRNU fingerprints of non-target cameras sets are available, we can then, use the pre-trained CNN on target cameras set so as to help improve generalization accuracy of the network for the new camera set.

5.5.2 Methodology

Figure 5.4 shows the overview of our proposed methodology for fine-tuned pre-trained CNN for source camera identification. In order to have a source problem that is close to the target problem, the proposed CNN model in Figure 5.2 was trained on a non-target camera classes. The PRNU images of the non-targeted cameras are used as input to the CNN model and trained. The trained model was saved. The trained CNN model is referred to as the pre-trained CNN model in Figure 5.4. The PRUN images of the targeted classes are then given as input to the pre-trained CNN model. Before fine-tuning of the pre-trained CNN model, its output layer (softmax layer) was removed before FC layers are added on top of the pre-trained network during the fine-tuning process. Also, all the weights of the layers of the pre-trained CNN model are updated during training. This means none of the weights of the layers of the pre-trained CNN are frozen. The output of the FC2 was given to a softmax classifier for probabilistic prediction of camera classes. Also, the fine-tuned model was used as feature extractor for one-vs-all linear SVMs. The feature for one-vs-all linear SVMs was extracted from the fine-tuned model prior to the addition of fully connected layers. The neurons in FC1 and FC2 are used as 128 and 256 respectively. However, we have to randomise the added fully connected layers (FC1 and FC2). This is because, the pre-trained CNN model will be eventually used for classifying the target camera classes and hence, the weights in the fully connected (FC) layers of the pre-trained CNN (CNN model trained for classifying the non-target classes) will not be suitable. The randomisation was achieved by using random initialisation in each fully connected layer. The training and the fine-tuning process of a CNN model have been described in Section 5.3.3. For the fine-tuned model, the MBSGD with momentum and decay were used for network optimisation. For a better generalisation of the network, 10-fold cross-validation was performed during fine-tuning.

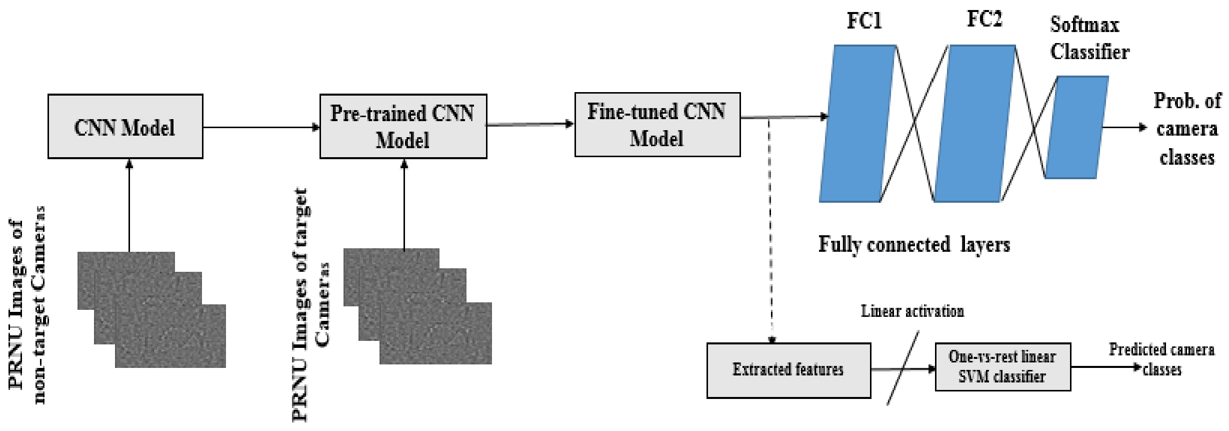


Figure 5.4. Overview of the Fine-tuned Pre-trained CNN.

5.5.3 Experiments and Results

The first 10 cameras in Table 4.1 are used as the non-target camera classes and the remaining 10 cameras are used as the target camera classes. The center part of the images was cropped to produce 64×64 image sizes. Optimal performance was obtained by using the same fine-tuning parameters used for the pre-trained CNN model. As shown in Table 5.9, fine-tuning the pre-trained CNN using CNN-SC shows 25.02% improvement over the identification accuracy of the proposed CNN model without fine-tuning. Likewise, fine-tuning the pre-trained model using CNN-SVM shows 20.37% improvement over the identification accuracy of the proposed CNN model without fine-tuning. Therefore, implementing a pre-trained CNN model already used on non-target camera classes can help improve the identification accuracy when used on target camera classes.

The confusion matrix for the 10 target camera classes using CNN-SC and CNN-SVM are shown in Table 5.10 and Table 5.11. The individual camera identification accuracy is shown by the diagonals of Table 5.3 in bold font. The average identification accuracy for both CNN-SC and CNN_SVM are 97.04 % and 96.82% respectively.

Table 5.9. Overall accuracies of proposed methods on target camera classes (%).

Proposed method with and without fine-tuning	**Target camera class**
CNN_SC without fine-tuning	71.9
CNN_SC with fine-tuning	96.92
CNN_SVM without fine-tuning	76.73
CNN_SVM with fine-tuning	97.10

Table 5.10. Identification accuracy (in percentage %) for 64 × 64 image size using CNN_SC.

Camera Device		11	12	13	14	15	16	17	18	19	20
Canon_Ixus70_0	11	**94.74**	2.92	1.75	0.58	-	-	-	-	-	-
Canon_Ixus70_1	12	1.12	**98.88**	-	-	-	-	-	-	-	-
Canon_Ixus70_2	13	1.75	1.75	**96.49**	-	-	-	-	-	-	-
Samsung_L74wide_0	14	-	-	-	**98.25**	0.87	0.87	-	-	-	-
Samsung_L74wide_1	15	0.45	-	-	0.45	**98.21**	-	-	-	0.89	-
Samsung_L74wide_2	16	-	-	-	2.60	3.46	**93.51**	-	0.43	-	-
Samsung_NV15_0	17	0.46	0.46	-	0.92	-	-	**96.31**	1.84	-	-
Samsung_NV15_1	18	-	-	-	-	-	-	1.87	**97.66**	0.47	-
Sony_DSC-H50_0	19	0.38	-	-	-	-	-	0.75	-	**98.87**	-
Sony_DSC-H50_1	20	-	0.43	0.85	-	-	-	-	-	1.28	**97.44**

Table 5.11. Identification accuracy (in percentage %) for 64 × 64 image size using CNN_SVM.

Camera Device		11	12	13	14	15	16	17	18	19	20
Canon_Ixus70_0	11	**94.15**	1.75	2.92	0.58	-	-	-	-	-	0.58
Canon_Ixus70_1	12	2.23	**96.09**	1.68	-	-	-	-	-	-	-
Canon_Ixus70_2	13	1.75	0.58	**97.08**	-	-	-	-	0.58	-	-
Samsung_L74wide_0	14	-	-	-	**98.25**	0.44	1.31	-	-	-	-
Samsung_L74wide_1	15	0.45	-	-	1.34	**98.21**	0.00	-	-	-	-
Samsung_L74wide_2	16	-	-	-	3.46	3.46	**93.07**	-	-	-	-
Samsung_NV15_0	17	-	-	-	-	-	0.46	**97.70**	1.84	-	-
Samsung_NV15_1	18	-	-	-	-	-	-	1.87	**97.66**	0.47	-
Sony_DSC-H50_0	19	-	-	-	-	-	-	0.75	0.38	**98.12**	0.75
Sony_DSC-H50_1	20	-	-	-	-	-	-	0.43	-	1.71	**97.86**

5.5.4 Comparison with Some State-of-the-Art Methods

Furthermore, we also compared our best system (CNN-SVM with fine-tuning) with four state-of-the-art methods for instance-based SCI for both 64×64 size on the target cameras. We compared with the same four state-of-the-art method in Section 5.4.3. Table 1 shows that the proposed method (CNN-SVM with fine-tuning) has better identification accuracy for each camera than all the compared state-of the arts methods. Also, our experimental results show that our proposed method using fine-tuned pre-trained convolutional neural networks for source camera identification has 24.03-38.52% identification accuracy higher than the compared four state-of-arts methods for 64 by 64 image size.

Table 5.12. Comparative study of the identification accuracies (in percentage %) of CNN-SVM with fine-tuning and four other state-of-the-art methods for 64×64 image sizes.

S/N	**Camera Device**	**MLE**	**Li's Model 5**	**Phase SPN**	**WA**	**CNN-SVM**
1	Canon_Ixus70_0	82.86	71.43	82.86	45.71	**94.15**
2	Canon_Ixus70_1	63.89	61.11	63.89	55.56	**96.09**
3	Canon_Ixus70_2	65.71	80.00	80.00	74.23	**97.08**
4	Samsung_L74wide_0	45.65	34.78	63.04	60.87	**98.25**
5	Samsung_L74wide_1	24.44	24.44	51.11	44.44	**98.21**
6	Samsung_L74wide_2	36.17	29.79	63.83	46.81	**93.07**
7	Samsung_NV15_0	61.36	38.64	61.36	50.00	**97.70**
8	Samsung_NV15_1	79.07	74.42	79.07	79.07	**97.66**
9	Sony_DSC-H50_0	85.19	83.33	90.74	83.33	**98.18**
10	Sony_DSC-H50_1	82.98	85.11	93.62	91.49	**97.86**
Average accuracy per camera device		62.73	58.31	72.95	63.15	**96.83**

5.6. Comparison with a Deep Learning Based Method

All four compared state-of-the-arts methods in Table 5.8 and Table 5.13 are PRNU-based methods. Therefore, we further compared our proposed CNN based methods with the work in [57]. The work proposes a deep learning-based method using content-adaptive fusion network (CA-FRN) for SCI. Their deep network learns camera features directly from the images while our proposed CNN-based methods learn from noise residues preprocessed by zero-meaning. The authors in [57] carried out experiments on three levels of camera identification which includes, camera brand identification (CBI), camera model identification (CMI) and camera device identification (CDI). A total of 13 camera devices from the Dresden database using 64×64 image size are used for their experimental evaluation. Therefore, in our comparison with CA-FRN, we also used the same camera devices and image size (64×64). Table 5.13 shows the list of cameras used to evaluate the proposed CA-FRN method. The differences in our evaluation settings include, the use of several patches of the same image for CA-FRN while we used only the PRNU of the center patch per image. The implication is that their network requires very huge training data for it to achieve good identification accuracy and hence the basis of the comparison is to see how our proposed CNN-SVM compares with CA-FRN in identification accuracy with and without pre-training with only few images for training our proposed deep networks. Hence, in our experiments, 80% of the camera images are used for training while the remaining 20% is used for testing.

Table 5.13. List of cameras used to evaluate the proposed CA_FRN for SCI.

S/N	Camera Brand	Resolution	Number of Images
1	Kodak_M1063_0	3664 × 2748	464
2	Pentax_optionA40_0	3000 × 4000	168
3	Nikon_CoolPixS710_1	3264 ×4532	197
4	Sony_DSC-H50_0	2736 × 3648	266
5	Olympus_mju_1050SW_2	3264 × 2736	218
6	Panasonic_DMC_FZ50_1	3684 × 2736	415
7	Agfa_Sensor530s_0	4032 × 3024	372
8	Ricoh_GX100_0	2736 × 3648	192
9	Samsung_NV15_0	2304 × 3072	217
10	Sony_DSC_W170_0	2736 × 3648	205
11	Sony_DSC_T77_0	2736 × 3648	181
12	Sony_DSC_T77_1	2736 × 3648	171
13	Sony_DSC_T77_2	2736 × 3648	189
Total number of Images			**3255**

Five of the cameras used in Table 5.1 are also part of the cameras used in Table 5.13 for evaluating their proposed CA_FRN. Therefore, for our proposed CNN-SVM with fine-tuning, we pre-trained on the remaining 15 cameras in Table 5.1 excluding the five cameras in Table 5.13. These 15 cameras are used as the non-target cameras as described in our methodology in Section 5.5.2 while the cameras in Table 5.13 are the target cameras. The work in [57] used the first 9 cameras in Table 5.13 for CBI, three model devices of Sony_DSC (cameras 4,10 and 11) for CMI and 3 devices of Sony_DSC_T77 (cameras 11-13) for CDI. The same settings are used also in our experimental evaluation for CBI, CMI and CDI.

Table 5.14. Comparative Study of the identification accuracies (%) of our proposed CNN-SVM with CA_FRN for Camera Brand Identification.

S/N	**Camera Device**	**CA-FRN**	**CNN-SVM1**	**CNN-SVM2**
1	Kodak_M1063_0	99.57	89.81	99.07
2	Pentax_optionA40_0	94.46	90.00	95.00
3	Nikon_CoolPixS710_1	97.49	69.44	90.00
4	Sony_DSC-H50_0	95.11	90.00	94.29
5	Olympus_mju_1050SW_2	97.65	91.42	97.75
6	Panasonic_DMC_FZ50_1	98.19	97.78	96.88
7	Agfa_Sensor530s_0	98.15	95.77	98.59
8	Ricoh_GX100_0	96.95	79.49	71.79
9	Samsung_NV15_0	96.95	81.81	77.27
Average		**97.8**	**87.28**	**91.18**

The comparison results are shown in Table 5.14, Table 5.15, and Table 5.16 for CBI, CMI and CDI respectively. CNN-SVM without and with fine-tuning are denoted as CNN-SVM1 and CNN-SVM2 respectively. The proposed CA-FRN [57] provides individual identification accuracy for the CBI while only average identification accuracies are provided for both CMI and CDI. Therefore, Table 5.15, and Table 5.16 only provide the average identification accuracies for CMI and CDI. For CBI, the average identification accuracies are 97.3%, 87.28%, 91.18% for CA-FRN, CNN-SVM without and with fine-tuning respectively. This shows CA-FRN is 10.02% and 6.12% greater in identification accuracies than our proposed CNN-SVM without and with fine-tuning for CBI. However, with just a single patch per image and less complexity of the network, our proposed CNN-SVM with fine-tuning has close identification accuracy with CA-FRN method.

Table 5.15. Comparative Study of the identification accuracies (%) of our proposed CNN-SVM with CA_FRN for Camera Model Identification.

S/N	**Camera Device**	**CA_FRN**	**CNN_SVM(Without fine-tuning)**	**CNN_SVM(With fine-tuning)**
4	Sony_DSC-H50_0	_	90.38	92.30
10	Sony_DSC_W170_0	_	79.07	72.09
11	Sony_DSC_T77_0	_	97.22	97.22
Average		**87.55**	**88.89**	**87.20**

For CMI, the average identification accuracies are 87.55%, 88.89%, and 87.20% for CA-FRN, CNN-SVM without and with fine-tuning respectively. Our proposed CNN-SVM methods have comparable identification accuracies with CA-FRN and our proposed CNN-SVM without fine-tuning is 1.34% greater in identification accuracy than CA-FRN. For CDI, the average identification accuracies are 73.27%, 86.10%, and 84.29% for CA-FRN, CNN-SVM without and with fine-tuning respectively. This shows that, our proposed CNN-SVM without and with fine-tuning have identification accuracies of 12.83% and 11.02% greater than CA-FRN method respectively. This shows that CA-FRN degrades highly in performance for CDI even with the use of several patches for training. CA-FRN extracts the camera features directly from the images and hence, it will require more features or image patches to learn distinct features compared to our approach. Our proposed CNN-based methods extract robust features directly preprocessed noise residues which are less contaminated by the scene contents of the images of cameras. Hence lesser number of images are expected to extract discriminative features.

Table 5.16. Comparative Study of the identification accuracies (%) of our proposed CNN-SVM with CA_FRN for Camera Device Identification.

S/N	**Camera Device**	**CA-FRN**	**CNN-SVM1**	**CNN-SVM2**
11	Sony_DSC_T77_0	_	84.21	76.31
12	Sony_DSC_T77_1	_	85.71	92.85
13	Sony_DSC_T77_2	_	88.37	83.72
Average		**73.27**	**86.10**	**84.29**

5.7 Robustness of Proposed CNN methods to Post JPEG Compression

A more challenging scenario for instance-based SCI is the determination of source cameras of images that have under-gone post-processing operations. There are several geometric distortions on images that affect the quality of PRNU fingerprints of cameras and common examples include cropping and post JPEG compression. In our PRNU extraction and formulation for proposed CNN method described in Section 5.3.1, camera images are already centered-cropped and hence our proposed CNN methods are robust to cropping. The sensor pattern noise has been previously reported to be robust to the compression due to internal processing during capturing of images by a camera [8]. However, when images are further compressed through sharing platforms such as WhatsApp, Facebook and others, the quantization noise due to compression can further suppress the content of the extracted PRNU fingerprints of cameras. This further reduction in the quality of the PRNU fingerprints due to post JPEG compression is one of the major limitation of using PRNU as a camera fingerprint. Therefore, in this section, we further examined the robustness of our proposed CNN to JPEG compression using our best CNN-system (CNN-SVM without and with fine-tuning). Since, most of the images shared on social platforms are usually natural images, then, we carried out this experiment using the natural images of the 10 cameras in case 2 only. The images of the cameras are compressed with the quality factors of 95, 90 and 80 before the extraction of 64×64 PRNU images. All other settings remained the same and 10-fold cross-validation was used. Table 5.17 shows the identification accuracies for each camera device without and with compression for CNN-SVM1 and CNN-SVM2 (CNN-SVM without and with fine-tuning) using quality factors of 95, 90, and 80 respectively. Table 5.17 shows, for CNN-SVM1, the average identification accuracies with post JPEG compression are 6.75%, 17.72% and 36.54% lesser than the average identification accuracy without post JPEG compression for quality factors of 95, 90 and 80 respectively. This shows that post JPEG compression has significant effect on our proposed CNN-system if not fine-tuned on CNN already trained on PRNU images of non-target cameras.

The decrease in accuracy can be attributed to reduction in the quality of the PRNU images due to post JPEG compression. However, for CNN-SVM2, the average identification accuracies with post JPEG compression are 0.68%, 1.74% and 3.37% lesser than the average identification accuracy without post JPEG compression for quality factors of 95, 90 and 80 respectively. This shows that our proposed CNN-system with fine-tuning using PRNU images as input, is robust to post JPEG compression with only little reduction in accuracy as compared to when the images are not compressed. CNN-SVM2 is robust to JPEG compression unlike CNN-SVM1 because, the pre-trained CNN used in CNN-SVM2 was trained on PRNU images extracted from images of non-target cameras without post JPEG compression.

Table 5.17. Identification accuracies of cameras in case 2 without and with JPEG Compression (%).

S/N	Without JPEG Compression		With JPEG Compression					
			QF=95		QF=90		QF=80	
	CNN-SVM1	**CNN-SVM2**	**CNN-SVM1**	**CNN-SVM2**	**CNN-SVM1**	**CNN-SVM2**	**CNN-SVM1**	**CNN-SVM2**
1	69.00	96.49	59.06	93.57	44.44	92.40	28.07	90.06
2	72.07	98.88	65.92	96.65	47.49	94.44	29.61	92.74
3	69.00	94.15	68.42	94.74	40.94	92.30	25.73	89.47
4	68.12	95.20	56.33	96.07	54.59	93.89	31.88	93.01
5	57.14	97.77	46.43	95.09	42..41	95.98	23.21	95.09
6	51.94	93.07	50.65	92.21	45.45	91.77	24.24	90.90
7	76.50	96.77	66.82	97.24	52.07	94.93	32.72	92.63
8	72.90	96.26	58.88	96.26	53.27	96.63	42.57	93.93
9	92.48	98.12	91.73	98.50	79.32	97.74	65.04	98.12
10	84.62	97.01	82.05	96.58	65.39	96.15	45.30	94.02
Average	**71.38**	**96.37**	**64.63**	**95.69**	**53.66**	**94.63**	**34.84**	**93.00**

Therefore, there is positive influence on quality of the features maps of the target cameras since the weights of all the layers of the pre-trained CNN are updated during training. This means the effect of post JPEG compression on the quality of the PRNU fingerprints of cameras can be suppressed by fine-tuning on a pre-trained model for data with related probabilistic distribution. Our experimental finding using pre-trained CNN is consistent with the work in [128] which studied the "the Impact of Standard Image Compression Techniques on Performance of Image

Classification with a Convolutional Neural Network". The work in [128] shows that, by using a pre-trained CNN to classify compressed images (image compressed by a factor 7, 16, 40 for a JPEG, JPEG200 and an HEVC encoder), a correct classification can still be maintained by CNN with only minimal effect on accuracy.

5.8 Summary

Instead of directly extracting features from the camera images, we propose the use CNN to learn robust features about the sensor noise patterns (photo-response non-uniformity). For source camera identification for small size images, we proposed deep CNN for classification and also as a feature extractor for one-versus-rest SVM. The contribution of the proposed methods can be summarized as below:

(i) CNN structure that is suitable for instance-based source camera identification using the noise residues of cameras was proposed. (ii) The proposed methods (CNN_SVM and CNN_SC) have superior performance, especially for small size images as compared with some state-of-the-art methods. (iii) The proposed methods for source camera identification from a small image patch would help in problems such as forgery localization with minimal data size but yet highly reliable result.

The motivation for using the sensor noise is that the sensor noise is unique for each individual camera so that CNN can be trained to capture each camera's unique information. The proposed neural network consists of three convolutional layers and two fully connected layers. The output was given to the regularized softmax classifier for probabilistic prediction of camera classes. Also, after training of the proposed CNN model, the flattened output of the third convolutional layer with a linear activation was extracted and given as the embedded layer for one-vs-rest linear SVMs. Experimental results show the efficiency and the accuracy of the proposed deep CNN-based methods in comparison with some state-of-the-art methods. Moreso, we evaluated the robustness of proposed deep CNN methods to post JPEG compression and our experimental results show that post JPEG compression has a significant effect on our proposed CNN-system if not fine-tuned on CNN already trained on PRNU images of non-target cameras. However, in the case of fine-tuned pre-trained CNN model, there is only a slight reduction in identification accuracy in comparison with proposed method with no post JPEG compression. Hence, the effect of post JPEG compression on the quality of the PRNU fingerprints of cameras can be suppressed by fine-tuning on a pre-trained model for data with related probabilistic distribution.

Chapter 6 : Deep Residual Convolutional Neural Network with Curriculum Learning For Source Camera Identification

In Chapter 5, the proposed deep CNN model only consists of the depth of three convolutional layers and two fully connected layers. Further increase in depth of convolutional layers on the proposed deep CNN model suffers degradation of accuracy on the noise residues of cameras from the Dresden dataset. The works in [109, 129] reveal that network depth is crucial in achieving a network with high generalization capability. However, an obstacle to increasing depth of network layers is the problem of vanishing or exploding gradients [65]. By vanishing gradient, we mean, the gradients of the network loss function tend towards zero and this makes network optimisation becomes difficult. To address the problem of the degradation in accuracy with increasing convolutional layers, residual neural network (ResNet) was proposed by He et al [27]. This is motivated by the need to increase the depth of deep learning models without the problem of vanishing gradients. This can be achieved by adding the output of the previous layers to the next layers using residual or short connections. Unlike CNN which fits a desired underlying mapping, the idea of ResNet is to use a residual mapping. The two main advantages of using ResNet are that it generates more robust representational bottlenecks and also tackles the problem of vanishing gradients through the smooth flow of data between networks. Therefore, in order to benefit from more robust or discriminative features capable of increasing identification accuracies of cameras, suitable architecture of deep residual network (ResNet) is investigated for instance-based source camera identification (SCI) for the noise residues or PRNU images of cameras.

Furthermore, to train a deep neural network, a large amount of training data is required, in order to avoid overfitting. Therefore, effective training algorithms are important for deep neural networks to achieve good generalization power especially for problems having small data. Images of camera have different texture complexity. High-quality PRNU fingerprint is extracted from smooth images or images with less texture complexity compared to clustered or natural images. This is because the PRNU fingerprints of cameras are contaminated by the scene contents. Hence, the lesser the scene contents of images of cameras, the better the extracted PRNU fingerprints of cameras. Based on this observation, we propose the use of curriculum learning algorithms to train our proposed deep ResNet for instance-based source camera identification. The idea of curriculum learning (CL) is to train a system, which may be a student or a deep network, from simple concepts to hard concepts. This learning approach allows the system to train up from handling simple tasks to hard tasks. The use of curriculum learning can help improves the speed of global convergence during training and a better local minimum can be achieved [39].

The rest of Chapter 6 is organized as follows. Section 6.1 gives an overview of a residual learning. Section 6.2 discusses curriculum learning for neural networks. Section 6.3 describes the framework of the proposed ResNet with curriculum learning. Section 6.4 presents experimental evaluation on 10 cameras from the Dresden database. Section 6.5 summarises the work and its contributions.

6.1 Overview of Deep Residual Learning

A deep neural network learns the underlying mapping of a given data, x. Let $G(x)$ represents the mapping function. This mapping function is to be learned by a stack of either fully connected layers or convolutional layers. The work in [27] assumes that if $G(x)$ can learn from the stack of layers consisting of non-linear functions, then, it can also approximate their residual functions. Assuming that the dimensions of the input and output layers are the same, the residual function ($F(x)$) can be expressed as, $G(x) - x$. Similarly, the underlying mappin $G(x)$ can be expressed as a function of the residual mapping as, $F(x) + x$. The assumption is that, instead of directly stacking additional layers, the additional layers can be added as identity mappings. Hence, a deeper network with lower training loss can be trained. The degradation of accuracy in deep networks without residual mapping can be attributed to the network optimisation algorithms finding it difficult to approximates mappings due to stack of non-linear layers. However, this becomes easier with the use of residual mapping since it helps the optimisation algorithms pushes the weights of network layers towards zero. Residual mapping adds shortcut connections to previous layers. The shortcut connections (skipping of one or more layers) perform identity mapping and the outputs are added to previously stacked layers so as to precondition the problem at hand and helps increase the rate of global convergence. The experimental results in [27] show that the suitable preconditioning of the problem by residual mapping is due to small responses generated by the trained residual network. Residual connections do not increase the computational complexity of the network since it has adds no extra parameters. The only computation required, is the element-wise addition between $F(x)$ and x. Element wise addition has negligible computational cost and this further makes ResNet attractive in practice. The output vectors of a layer (y) after performing residual mapping on the input vectors of the previous layer (x) is expressed in [27] as,

$$y = F(x, \{W_i\}) + x \tag{6.1}$$

where W_i is the weight of the layer, i and $F(x, \{W_i\})$ is the learned residual mapping. Eqn. (6.1) assumes that the dimensions of F and x are equal. In a situation where the sizes of F and x are different, then, the size of the input x can be changed by using a linear transformation, P_s. Hence, eqn. (6.2) becomes,

$$y = F(x, \{W_i\}) + P_s x \tag{6.2}$$

The basic function of P_s is dimension matching. Examples of P_s can include the use of either a fully connected layer or a convolutional layer of kernel size 1×1 without the use of activation functions.

6.2 Curriculum Learning for Neural Networks

The idea of CL is motivated by the education system where learning is introduced from simple concepts to hard concepts. Organizing the education system this way helps the students to leverage hard concepts based on their understanding of the easy concepts. This same idea can be applied to the training of neural networks where training is initiated to begin on examples of the dataset which are easier to learn by a network before introducing examples with more complexity. In order words, CL works on a sequence of training sets via progressive training on smoothed data before the consideration of less smoothed data. In CL, the reweighting of examples in the training set is uniform as training progresses from simplest concepts to the target training set [39]. To further illustrates CL mathematically, consider a random variable, $q(x, y)$, where x and y are the training examples, and their corresponding labels respectively. At each step, τ in the curriculum sequence (i.e. the division of q into data subsets), let the corresponding weight be $0 \leq W_\tau(q) \leq 1$. $W_1(q) = 1$, $0 \leq \tau \leq 1$ and $Q(q)$ be the target training distribution. The training distribution $(P_\tau(q))$ of $Q(q)$ is expressed in [39] as,

$$P_\tau\,(q) \propto W_\tau(q)Q(q) \quad \forall q \tag{6.3}$$

Given that, $\int P_\tau(q)dq = 1$, then eqn. (6.3) becomes,

$$P_1(q) = W_1(q)Q(q) = Q(q) \quad \forall q \tag{6.4}$$

∀ is an existential quantifier and it means the eqns. (6.3) and (6.4) hold true for all instances of the random variable, q. Note that τ is a monotonically increasing step size. Eqn. (6.3) and eqn. (6.4) only satisfy CL provided that the entropy of each subset of q monotonically increases. Increase in entropy means, at each addition of a new data subset during training, the weights of the data subset also increases. Curriculum learning has demonstrated in [39] helps to improve the speed of global convergence during training and a better local minimum is achieved. It has also been used with deep CNN in many applications such as depth estimation [130], facial expression recognition [131], hand posture recognition [132], and localization of thoracic disease on chest radiographs [133]. What easy and hard examples mean depending on the area of the application. In facial expression recognition[131], those face images with a high expression intensity can be considered easy examples, while those images with a low expression intensity are hard examples.

6.3 Proposed Deep Residual Convolutional Neural Network and Curriculum Learning Algorithms for Source Camera Identification

The PRNU image in a smooth or flat image is easier to learn than that in a natural image, which contains more complex textures. In our CL algorithm, flat images are used as easy, simple samples, while natural images are used as hard examples. The general framework for our proposed algorithm is shown in Figure 6.1. The noise residues in the images extracted by a wavelet-based denoising method [5] form the PRNU images, which are arranged according to the complexity of the images, from easy images, i.e. those smooth and flat images, to difficult images, i.e. those natural images, and are divided into d groups, denoted as $(D_1, D_2 \ldots . D_d)$. In other words, d subsets are formed for curriculum learning. Assume that the images are generated by K cameras, i.e. there are K classes, and n_i is the number of samples in a subset D_i. Then, the total number of samples N in the camera dataset is given by $N = \sum_{i=1}^{d} n_i$. The denoising method applied to the camera images is based on a wavelet-based denoising filter [5], and the noise residues in an image captured by a specific camera are obtained by subtracting the filtered images from the original images. The noise residues in an image are also pre-processed by the zero-mean operation. This zero-mean operation is applied to row-by-row, followed by column-by-column, of each noise-residue image. This operation can help reduce the effect of linear patterns introduced into the noise residues, due to the color interpolation and pipeline processing operations of sensor and electronic circuits in cameras [134]. It also acts as a normalization process. Each training subset, D_1 to D_d, is used to train the ResNet sequentially. After all the subsets have been used for training, the features from the last convolutional layer of the trained ResNet are extracted to form the deep features of the input samples. These extracted deep features are then used to learn one-vs-rest linear support vector machines (SVMs) for predicting the camera classes $(C_1, C_2, \ldots . C_K)$. The use of the one-vs-rest linear SVM classifiers results in more training samples for the classifiers. The architecture of the proposed deep residual convolutional neural network is described in Section 6.3.1 while the details of the training procedures for the proposed manual and automatic curriculum learning algorithms are described in Section 6.3.2.

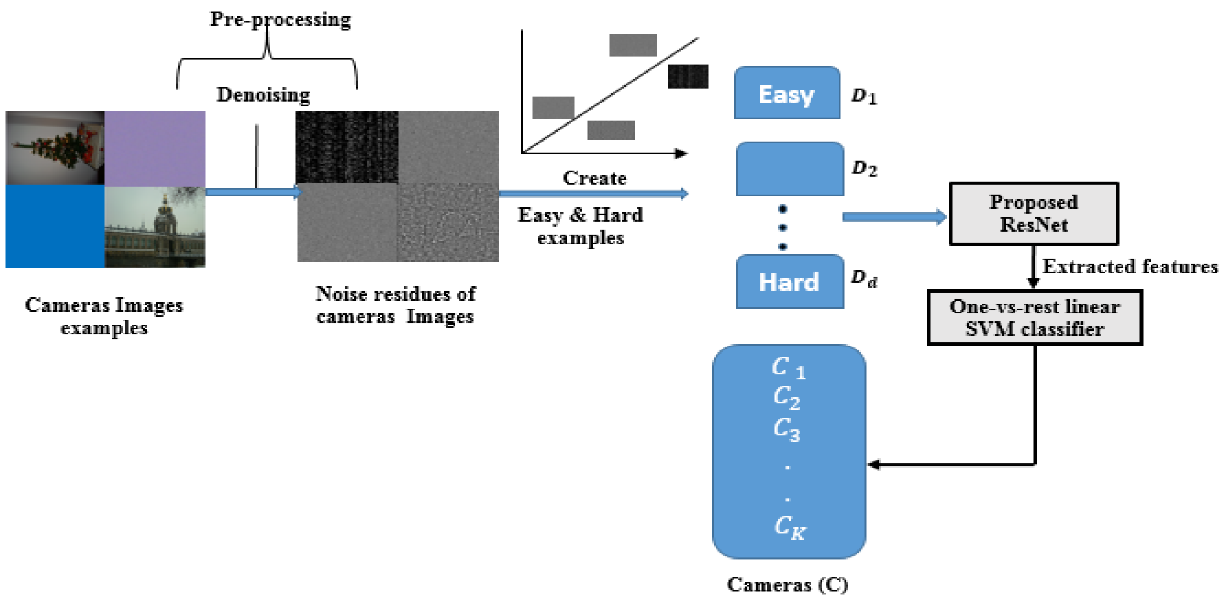


Figure 6.1. The general framework of our proposed deep residual convolutional neural network with curriculum learning for source camera identification.

6.3.1 Architecture of the Proposed Deep Residual Convolutional Neural Network

Figure 6.2 shows the layout of the proposed deep residual convolutional network for instance-based camera source identification.The main difference between the proposed architecture for deep CNN in Figure 5.2 and Figure 6.2 is the addition of residual connections and convolutional layers. Only the difference in architectures is explained as follows. The optimal performance with the proposed ResNet is achieved in our experiments with three residual neural connections and two convolutional added to our initially proposed deep CNN architecture. Since a stride of 2 × 2 is used in Conv5, Conv4 which is a 1×1 convolution operation is used to linearly down-sample the output of Conv3 so that it can have the same feature-map size when concatenated with Conv5 before given it as input to two fully connected layers FC1 and FC2.

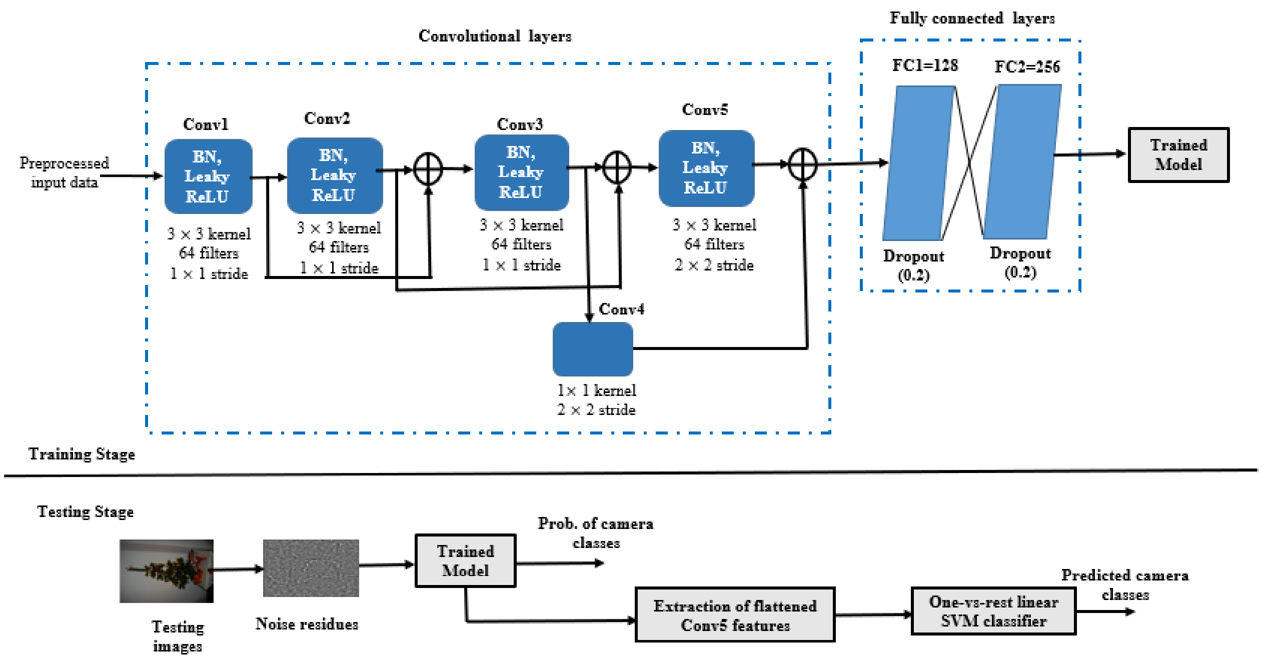


Figure 6.2. The layout of the proposed deep residual convolutional neural network for instance-based camera source identification.

Batch normalization and ReLU operations were not carried out in Conv4. The additional convolutional layers and residual connections beyond Conv5 do not lead to an increase in cameras identification accuracies. Apart from dropout regularization applied n FC1 and FC2 in Figure 6.2, sparsity constraint and weight regularization methods are used with the regularization parameter of 10^{-5} each in the FC1, FC2 and softmax layers to further prevent model overfitting. Imposing sparsity constraint is a form of regularization, but not weight regularization. It regularizes the outputs of the layer rather than the weight of the layer. Weight regularization can be achieved by using $L2$ norm regularization, while the sparsity constraint is imposed by adding the absolute values of the true value of a layer into the loss function. The same training and the fine-tuning process used for the proposed deep CNN methods in Chapter 5 are also used for the proposed ResNet model. Testing images are given to the trained ResNet model and probabilities of cameras are predicted using softmax classifier. We also extracted the features of the flattened layers of the Conv5. The extracted deep features are used to learn one-vs rest linear SVM for the prediction of camera classes.

6.3.2 Proposed Curriculum Learning Algorithms

In this section, we propose a manual and an automatic CL algorithm. For the manual CL algorithm, the easy and hard examples are selected manually prior to the training. For the automatic CL algorithm, the training data are sorted in the order of increasing complexity. The training procedures for the proposed manual and automatic CL algorithms are listed as follows.

Manual Curriculum Learning

i. The noise residues of flat images are first used to train the proposed ResNet model.

ii. The best-trained model is obtained.

iii. The best-trained ResNet model is trained with a decreased learning rate.

iv. The noise residues of natural images are now used to train the trained ResNet.

v. The best-trained model is then used for extracting deep features for camera identification.

The two data subsets are trained with 20 epochs. The learning rate is decreased as stated in (iii), so that the negative influence from the hard examples can be reduced. The learning rate is reduced from 0.001 to 0.0007.

Automatic Curriculum Learning

i. The input dataset contains all the training images (x) and the corresponding labels (y).
ii. The best version of ResNet is trained on the dataset (x, y).
iii. The predicted training features (X) of the softmax layer of the trained model are extracted.
iv. The softmax loss of X is then calculated. The softmax loss (p) can be expressed as,

$$p_i = \frac{e^{X^i}}{\sum_{j=1}^{K} e^{X^j}} \tag{6.5}$$

where $i = 1{:}N$, X^i is each training instance, K is number of camera classes, $j = 1{:}K$ and X^j is the value of X^i in j.

v. The cross-entropies for each instance in X are obtained. The cross-entropy loss can be defined as,

$$L_c(y, p_i) = -\sum_{i=1}^{N} y_i \log p_i \tag{6.6}$$

where y_i is the class label of a training instance, X^i. The value of y_i ranges from 1: K.

vi. Training instances with smaller cross-entropies can be better optimized than training instances with larger cross-entropies, hence, the cross-entropies are sorted in ascending order. The indices of the sorted cross-entropies are used to re-order the original dataset, (x, y).

vii. The re-ordered dataset, (x, y), is used to train the ResNet model, and the trained ResNet is used to predict the camera classes.

6.4 Experimental Evaluation and Discussion

A total of 10 cameras in Table 6.1 including both flat and natural images are used for experimental evaluation of the proposed methods. These are the same as the case 2 cameras in Table 5.1 (the last ten cameras). The center part of the images are cropped to produce 64×64 image sizes. The total number of flat and natural images are 2136 and 500 respectively. Flat images can only be used for training and not used as test data for the source camera identification problem. Similarly, as carried out with our earlier proposed deep CNN, we also used 10-fold used cross-validation during training for our experiments on the proposed ResNet. Table 6.2 shows the identification accuracies with and without fine-tuning for the CNN-SVM and ResNet-SVM respectively. CNN-SVM and ResNet-SVM are used to denote the identification accuracy for deep CNN with one-vs-rest SVM classifier and deep residual CNN with one-vs-rest SVM classifier respectively. The fine-tuning process is the same with the same proposed fine-tuned CNN approach in Section 5.5.2 of Chapter 5. Our fine-tuning approach involves using pre-trained deep CNN model on 10 non-target camera classes and fine-tuning it on 10 target camera classes. The average identification accuracies without fine-tuning for both CNN-SVM and ResNet-SVM are 75.67% and 77.85% respectively while the average identification accuracy for both CNN-SVM and ResNet-SVM are 96.83% and 97.10% respectively. This shows that ResNet-SVM has 2.18% and 0.27% higher identification accuracies than CNN-SVM without and with fine-tuning. Therefore, adding residual connections to our CNN architecture for source camera identification helps to improve cameras identification accuracies.

Table 6.1. List of Cameras Used.

S/N	Camera Brand	Resolution	Natural Images	Flat Images
1	Canon_Ixus70_0	2304 × 3072	171	50
2	Canon_Ixus70_1	2304 × 3072	179	50
3	Canon_Ixus70_2	2304 × 3072	171	50
4	Samsung_L74wide_0	2304 × 3072	229	50
5	Samsung_L74wide_1	2304 × 3072	224	50
6	Samsung_L74wide_2	2304 × 3072	231	50
7	Samsung_NV15_0	2304 × 3072	217	50
8	Samsung_NV15_1	2304 × 3072	214	50
9	Sony_DSC-H50_0	2736 × 3648	266	50
10	Sony_DSC-H50_1	2736 × 3648	234	50
Total number of Images			**2136**	**500**

For curriculum learning, several progressions are tested to determine the best progression of data subsets with the ResNet model. The tested progression of data subsets includes, no curriculum learning, curriculum learning and anti-curriculum learning. By anti-curriculum learning, we mean, training on hard examples before easy examples. In these experiments, the flat images are used only for training while 80% of the natural images are used for training and the remaining 20% for the evaluation of the best-trained ResNet model. Table 6.3 shows the identification accuracies for both softmax (ResNet-SC) and one-vs-rest linear SVM (ResNet-SVM). PRNU_F and PRNU_N denote the PRNU of flat and natural images respectively.

Table 6.2. Identification accuracies for the Proposed Deep CNN and RCNN (%).

Cameras	Without Fine-tuning		With fine-tuning	
	CNN-SVM	**ResNet-SVM**	**CNN-SVM**	**ResNet-SVM**
Canon_Ixus70_0	71.35	76.02	94.15	95.91
Canon_Ixus70_1	77.09	81.01	96.09	97.21
Canon_Ixus70_2	71.93	73.68	97.08	95.91
Samsung_L74wide_0	74.24	71.62	98.25	97. 38
Samsung_L74wide_1	63.84	68.75	98.21	95.54
Samsung_L74wide_2	66.23	67.97	93.07	96.54
Samsung_NV15_0	78.80	82.95	97.70	97.24
Samsung_NV15_1	74.77	76.17	97.66	97.20
Sony_DSC-H50_0	91.73	93.61	98.18	99.62
Sony_DSC-H50_1	86.75	86.75	97.86	98.72
Average	**75.67**	**77.85**	**96.83**	**97.10**

As shown in Table 6.3, the best performance is obtained by training the PRNU of the flat images first and then followed by the PRNU of both flat and natural images. Its overall identification accuracy using ResNet-SVM is 0.7% greater when PRNU of the flat images are trained followed by PRNU of natural images and 3.74% higher than training ResNet with no curriculum learning. Table 6.3 also shows that the progression involving anti-curriculum learning has lower identification accuracies compared to when there is no curriculum learning and with curriculum learning. The natural images are further divided into smooth, saturated and others as in [57] but experimental evaluation shows degradation on performance compared to when all natural images are used without division into further subsets. Also, we carried out experiment on automatic curriculum learning using the proposed algorithm in Section 6.3.2 with ResNet-SVM. It has 0.47% identification accuracy greater than training with no curriculum learning. This indicates that there is only little impact on the accuracy, with and without using the automatic CL.

Table 6.3. Overall accuracy for manual curriculum learning with combinations and orders of the flat images and natural images (%).

Curriculum Learning (CL)		
Different Progressions	**ResNet-SC**	**ResNet-SVM**
PRNU_F+ PRNU_N	66.82	73.36
PRNU_F, PRNU_F+ PRNU_N	67.76	**77.10**
PRNU_F+ PRNU_N, PRNU_F	50.71	73.83
PRNU_F, PRNU_N	**72. 43**	76.40
PRNU_N , PRNU_F	57.94	72.43
Automatic Learning		
PRNU_F+ PRNU_N	65.89	73.83

Therefore, using manual curriculum learning with ResNet for source camera identification has more impact on the identification accuracy compared to using the proposed automatic curriculum learning approach. Figure 6.3 shows the logarithm of the variation of the error rates or the model losses with respect to the number of epochs for without CL, with manual CL, automatic CL and anti-CL methods. As observed in Figure 6.3, the model becomes stable towards 20 epochs. The confusion matrix for the ten cameras, with the best trained ResNet model, is shown in Table 6.4.

The individual camera identification accuracies are shown in the diagonal of Table 6.4 and highlighted in bold.

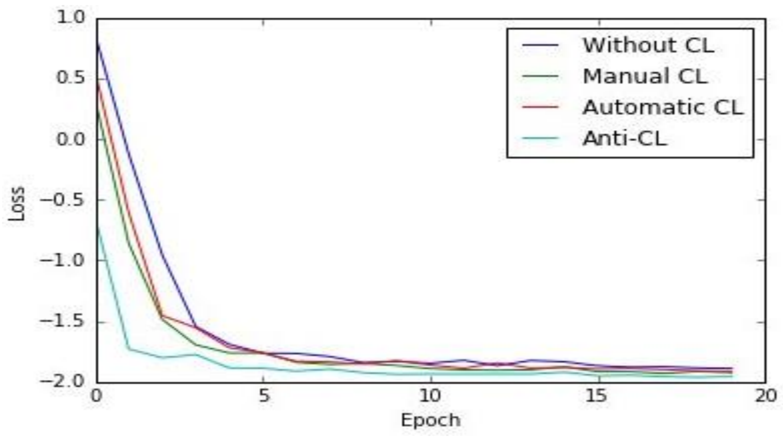


Figure 6.3. The Logarithm of model losses against epochs for without CL, manual CL, automatic CL and anti-CL methods.

Table 6.4. Identification accuracy (%) of the best result of the proposed method.

Camera Device		1	2	3	4	5	6	7	8	9	10
Canon_Ixus70_0	1	**76.32**	13.16	5.26	-	-	-	-	5.26	-	-
Canon_Ixus70_1	2	2.78	**83.33**	13.90	-	-	-	-	-	-	-
Canon_Ixus70_2	3	-	18.75	**78.12**	-	-	3.12	-	-	-	-
Samsung_L74wide_0	4	-	22.20	22.20	**64.44**	17.78	11.11	2.22	-	-	-
Samsung_L74wide_1	5	-	23.80	-	16.67	**61.90**	11.90	4.76	2.38	-	-
Samsung_L74wide_2	6	-	-	-	4.88	14.63	**75.61**	-	4.88	-	-
Samsung_NV15_0	7	-	-	-	-	-	-	**94.87**	5.13	-	-
Samsung_NV15_1	8	-	2.00	-	2.00	-	-	26.00	**68.00**	2.00	-
Sony_DSC-H50_0	9	-	-	-	-	-	-	-	-	**93.75**	6.25
Sony_DSC-H50_1	10	-	-	-	-	-	-	2.44	-	7.32	**90.24**

The average individual accuracy for the 10 cameras for images of size 64 × 64 is 78.66%. The same cameras and settings used for our experimental evaluation are also used in our comparison with the state-of-the-art methods in Section 5.5.4 of Chapter 5. The compared state-of-the-art methods are, MLE SPN [8], Phase SPN [10], the Li's model [9] and weighted averaging (WA) [51] and their overall average identification accuracies are, 62.73%, 72.95% ,58.31% and 63.15% respectively. We can see that the overall average individual camera accuracy of the proposed method is 15.93%, 5.71%, 20.35% and 15.51% higher than that of the MLE SPN, Phase SPN, Li's model and WA methods respectively. Hence, an accurately designed deep network, based on good training algorithms, can achieve better performance than the conventional or PRNU-based SCI methods, for small-query images using the same number of training and testing examples.

6. 5 Summary

Deep residual convolutional neural networks (ResNet) is proposed for source camera identification using noise residues or individual PRNU images of cameras. Experiments are conducted with and without fine-tuning on the pre-trained network. Experimental results showed the benefits of adding residual connections to our initially proposed deep CNN without residual connections. Experiments were conducted with different orders of the camera subsets (simple and hard subsets) from the Dresden database, with our manual and automatic curriculum learning algorithms. For the proposed manual curriculum learning, experimental results show that training based on easy training examples before hard examples will result in the best identification accuracy, based on the proposed ResNet model. Furthermore, our proposed automatic curriculum learning approach shows better identification accuracy compared to training without applying curriculum learning. In conclusion, our proposed deep learning methods for instance-based SCI can achieve better performance than the compared state-of-the-art methods using the same settings.

Chapter 7 :

Conclusion and Future Works

7.1 Conclusion of the Thesis

In this thesis, aiming at using deep learning due to its discriminative power to achieve higher camera identification accuracy, we proposed two deep learning techniques for instance-based source camera identification especially tailored towards small sizes images. This work can then be deployed in applications involving splicing localization and also source camera identification for small-sized image forgeries.

Firstly, we proposed supervised stacked sparse autoencoder to extract distinctive features from the noise residue of cameras. Experimental results show that the proposed method has comparable significant identification accuracy on the Dresden database and better identification accuracy on our dataset consisting of two phone cameras. Further experiments also show that with better quality PRNU fingerprints, the proposed SSAE has superior performance than some of the state-of-the-art methods. Also, our SSAE generalizes well on new cameras' set using the same parameters used on another set of cameras.

Secondly, we proposed the use of deep CNN to solve instance-based source camera identification using noise residues of cameras. The proposed CNN was used for classification (CNN-SC) and also as a feature extractor (CNN-SVM). Our experimental results show that the proposed CNN-SVM performs better than CNN-SC in all experiments. Also, we explored the advantage of fine-tuning pre-trained models. In our approach, we fine-tuned already pre-trained proposed CNN model. Our experimental results show the effectiveness of our proposed fine-tuned deep CNN over deep CNN model without fine-tuning. Also, a comparative study with some state-of-the-art methods is carried out. Experimental results show that the identification accuracies of our proposed CNN-based methods (CNN-SC and CNN-SVM) are 18%-25.6% and 20.37%-25.02% higher than four compared PRNU-based SCI methods for without and with fine-tuning on a pre-trained deep CNN model. We also compared with a deep learning-based method, content-adaptive fusion networks (CA-FRN) and our proposed CNN-based methods (CNN-SVM without and with fine-tuning) are 6.12-10.02% lesser in identification accuracy than CA-FRN for camera brand identification but have identification accuracies of 1.34% and 11.02-12.83% greater than CA-FRN for camera model Identification and camera device identification respectively.

Moreso, the proposed deep CNN methods are also evaluated under post JPEG compression of the images. The effect of post JPEG compression on the quality of the PRNU fingerprints of cameras can be suppressed by fine-tuning on a pre-trained model while post JPEG compression has a significant effect on our proposed CNN-system if not fine-tuned on CNN already trained on PRNU images of non-target cameras. The proposed deep CNN methods are evaluated under post JPEG compression of the images. The effect of post JPEG compression on the quality of the PRNU fingerprints of cameras can be suppressed by fine-tuning on a pre-trained model while post JPEG compression has a significant effect on our proposed CNN-system if not fine-tuned on CNN already trained on PRNU images of non-target cameras.

Finally, since the entropy of natural images is higher than that of flat images of cameras, we proposed the use of CL with ResNet for source camera identification. From our experimental results, both proposed ResNet and CL algorithms improve the identification accuracies of cameras compared to the proposed deep CNN methods for SCI.

7.2 Future Works

Despite some of our achievement in using SSAE for source camera identification, there are still some limitations regarding the proposed method. The major limitation of using SSAE is that the camera identification accuracy only has comparable results with the state of the art methods when natural images were used only for training and not flat images. This limitation could be due to the amount of training data used in our implementation since better generalization accuracy are usually obtained using huge data for most deep learning techniques. Also, another challenge could be attributed to the quality of the noise residue used. This is because, single noise residue representing image will have scene contamination and hence finding ways of improving the quality of the PRNU could be a better way of reaching global convergence quickly during training. Future research should consider an approach that can improve PRNU detection accuracy particularly, the transformation and preprocessing techniques to be able to increase the detection capability of the proposed method. Furthermore, fine-tuning techniques for our proposed SSAE and as well as other variants of SAE can also be explored.

Finally, though, using the deep convolutional neural network for source camera identification, we were able to achieve a substantial result as compared to existing methods but there is still some drawback to attain the desired accuracy for the CNN structure proposed.

Future research should consider ways to reduce the validation error and also techniques that can improve network generalization accuracy. Furthermore, we will also explore featuring engineering techniques that could make the input noise images invariant to the original images. Also we shall explore suitable network engineering techniques with CNN that can help achieve better camera identification accuracy. Finally, as regards, CL with ResNet for SCI, future research could consider using suitable pre-processing operations to generate simple examples from the original training set so as to further increase the number of training subsets. Moreover, a better automatic curriculum learning approach will be explored, so that the learning efficiency can be improved, and hence increase the camera detection accuracies.

References

[1] Y. Hu, B. Yu, and C. Jian, "Source camera identification using large components of sensor pattern noise," in *2nd International Conference on Computer Science and its Applications (CSA'09)*, 2009, pp. 1-5.

[2] H. T. Sencar and N. Memon, "Overview of state-of-the-art in digital image forensics," in *WSPC - Proceedings on Algorithms, Architectures and Information Systems Security*, September 2007, vol. 3, pp. 325-348.

[3] A. C. Kot and H. Cao, "Image and video source class identification," in *Digital Image Forensics*. New York: Springer, 2013, pp. 157-178.

[4] P. Blythe and J. Fridrich, "Secure digital camera," in *Digital Forensic Research Workshop*, 2004, pp. 11-13.

[5] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security,* vol. 1, no. 2, pp. 205-214, June 2006.

[6] I. Amerini, R. Caldelli, V. Cappellini, F. Picchioni, and A. Piva, "Estimate of PRNU noise based on different noise models for source camera identification," in *Crime Prevention Technologies and Applications for Advancing Criminal Investigation*. USA: IGI Global, 2012, pp. 9-20.

[7] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security,* vol. 1, no. 2, pp. 205-214, June 2006.

[8] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, "Determining image origin and integrity using sensor noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74-90, March 2008.

[9] C. T. Li, "Source camera identification using enhanced sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. p. 280-287, June 2010.

[10] X. Kang, Y. Li, Z. Qu, and H. J., "Enhancing source camera identification performance with a camera reference phase sensor pattern noise," *IEEE Transactions on Information Forensics and Security,* vol. 7, no. 2, pp. 393-402, April 2012.

[11] X. Lin and C.-T. Li, "Preprocessing reference sensor pattern noise via spectrum equalization," *IEEE Transactions on Information Forensics and Security,* vol. 11, no. 1, pp. 126-140, 2016.

[12] A. Lawgaly and F. Khelifi, "Sensor Pattern Noise Estimation Based on Improved Locally Adaptive DCT Filtering and Weighted Averaging for Source Camera Identification and Verification," *IEEE Transactions on Information Forensics and Security,* vol. 12, no. 2, pp. 392-404, 2016.

[13] A. J. Cooper, "Improved photo response non-uniformity (PRNU) based source camera identification," *Forensic science international,* vol. 226, no. 1, pp. 132-141, 2013.

[14] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, "Compressed fingerprint matching and camera identification via random projections," *IEEE Transactions on Information Forensics and Security,* vol. 10, no. 7, pp. 1472-1485, 2015.

[15] Y. Tomioka, Y. Ito, and H. Kitazawa, "Robust digital camera identification based on pairwise magnitude relations of clustered sensor pattern noise," *IEEE transactions on information forensics and security,* vol. 8, no. 12, pp. 1986-1995, 2013.

[16] J. Salmon, Z. Harmany, C.-A. Deledalle, and R. Willett, "Poisson noise reduction with non-local PCA," *Journal of mathematical imaging and vision,* vol. 48, no. 2, pp. 279-294, 2014.

[17] Y. Huang, J. Zhang, and H. Huang, "Camera model identification with unknown models," *Information Forensics and Security, IEEE Transactions on,* vol. 10, no. 12, pp. 2692-2704, 2015.

[18] M. Kharrazi, H. T. Sencar, and N. Memon, "Blind source camera identification," in *International Conference on Image Processing , ICIP'04.*, 2004, vol. 1: IEEE, pp. 709-712.

[19] F. Ahmed, F. Khelifi, A. Lawgalv, and A. Bouridane, "Comparative Analysis of a Deep Convolutional Neural Network for Source Camera Identification," in *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3),* 2019: IEEE, pp. 1-6.

[20] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single sample face recognition via learning deep supervised autoencoders," *IEEE Transactions on Information Forensics and Security,* vol. 10, no. 10, pp. 2108-2118, October 2015.

[21] Y. Lu, L. Zhang, B. Wang, and J. Yang, "Feature ensemble learning based on sparse autoencoders for image classification," in *2014 International Joint Conference on Neural Networks (IJCNN),* July 2014: IEEE, pp. 1739-1745.

[22] R. F. Nogueira, R. de Alencar Lotufo, and R. C. Machado, "Fingerprint liveness detection using convolutional neural networks," *IEEE Transactions on Information Forensics and Security,* vol. 11, no. 6, pp. 1206-1213, March 2016.

[23] O. E. David and N. S. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," in *2015 International Joint Conference on Neural Networks (IJCNN),* 2015: IEEE, pp. 1-8.

[24] F. N. Khan, K. Zhong, W. Al-Arashi, C. Yu, C. Lu, and A. P. T. Lau, "Modulation format identification in coherent receivers using deep machine learning," *IEEE Photonics Technology Letters,* vol. 28, no. 17, pp. 1-4, June 2016.

[25] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *Journal of Sensors,* vol. 2016, no. 2016, pp. 1-10, June 2015.

[26] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory dsp]," *Signal Processing Magazine, IEEE,* vol. 28, no. 1, pp. 145-154, 2011.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[28] L. Bondi, L. Baroffio, D. Guera, P. Bestagini, E. J. Delp, and S. Tubaro, "First Steps Towards Camera Model Identification with Convolutional Neural Networks," *IEEE Signal Processing Letters,* pp. 1-5, 2016.

[29] L. Baroffio, L. Bondi, P. Bestagini, and S. Tubaro, "Camera identification with deep convolutional networks," *arXiv preprint arXiv:1603.01068,* 2016.

[30] 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.

[31] A. Budiman, M. I. Fanany, and C. Basaruddin, "Stacked denoising autoencoder for feature representation learning in pose-based action recognition," in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, October 2014: IEEE, pp. 684-688.

[32] D. Luo, R. Yang, B. Li, and J. Huang, "Detection of Double Compressed AMR Audio Using Stacked Autoencoder," *IEEE Transactions on Information Forensics and Security,* vol. 12, no. 2, pp. 432-444, Febuary 2017.

[33] W. W. Ng, G. Zeng, J. Zhang, D. S. Yeung, and W. Pedrycz, "Dual autoencoders features for imbalance classification problem," *Pattern Recognition,* vol. 60, pp. 875-889, June 2016.

[34] A. Ng. "CS294A lecture notes on sparse autoencoder." https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf (accessed.

[35] F. Chollet. "Building autoencoders in Keras." https://blog.keras.io/building-autoencoders-in-keras.html (accessed.

[36] B. Bayar and M. 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*, 2016: ACM, pp. 5-10.

[37] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849-1853, 2015.

[38] A. Tuama, F. Comby, and M. Chaumont, "Camera Model Identification With The Use of Deep Convolutional Neural Networks," in *IEEE International Workshop on Information Forensics and Security*, 2016, pp. 1-6.

[39] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009: ACM, pp. 41-48.

[40] H. R. Chennamma and L. Rangarajan, "Source camera identification based on sensor readout noise," in *Crime Prevention Technologies and Applications for Advancing Criminal Investigation*. USA: IGI Global, 2012, pp. 21-34.

[41] K. Choi, E. Y. Lam, and K. Wong, "Automatic source camera identification using the intrinsic lens radial distortion," *Optics Express*, vol. 14, no. 24, pp. 11551-11565, 2006.

[42] L. T. Van, S. Emmanuel, and M. S. Kankanhalli, "Identifying source cell phone using chromatic aberration," in *IEEE International Conference on Multimedia and Expo*, 2007: IEEE, pp. 883-886.

[43] A. C. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948-3959, October 2005.

[44] S. Bayram, H. T. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," in *IEEE International Conference on Image Processing, ICIP*, 2005, vol. 3: IEEE, pp. III-69.

[45] A. Swaminathan, M. Wu, and K. J. Liu, "Nonintrusive component forensics of visual sensors using output images," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 91-106, March 2007.

[46] H. Cao and A. C. Kot, "Similar DSLR processor identification using compact model Template," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC'11)*, 2011, pp. 1-12.

[47] J. Fridrich, "Sensor defects in digital image forensic," in *Digital Image Forensics*: Springer, 2013, pp. 179-218.

[48] Y. Hu, C. Jian, and C. T. Li, "Using improved imaging sensor pattern noise for source camera identification," in *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, 2010: IEEE, pp. 1481-1486.

[49] A. Lawgaly, F. Khelifi, and A. Bouridane, "Image sharpening for efficient source camera identification based on sensor pattern noise estimation," in *2013 Fourth International Conference on Emerging Security Technologies (EST)*, 2013: IEEE, pp. 113-116.

[50] M. Chen, J. Fridrich, and M. Goljan, "Digital Imaging Sensor Identification (further study)," in *Electronic Imaging*, Jan. 2007: International Society for Optics and Photonics, pp. 65050P-65050P-13.

[51] A. Lawgaly, F. Khelifi, and A. Bouridane, "Weighted averaging-based sensor pattern noise estimation for source camera identification," in *Image Processing (ICIP), 2014 IEEE International Conference on*, 2014: IEEE, pp. 5357-5361.

[52] T. Gloe, S. Pfennig, and M. Kirchner, "Unexpected artefacts in PRNU-based camera identification: a'Dresden Image Database'case-study," in *Proceedings of the on Multimedia and security*, 2012: ACM, pp. 109-114.

[53] H. Zeng, Y. Wan, K. Deng, and A. Peng, "Source Camera Identification With Dual-Tree Complex Wavelet Transform," *IEEE Access*, vol. 8, pp. 18874-18883, 2020.

[54] M. Zhao, B. Wang, F. Wei, M. Zhu, and X. Sui, "Source camera identification based on coupling coding and adaptive filter," *IEEE Access*, 2019.

[55] F. Bellavia, M. Iuliani, M. Fanfani, C. Colombo, and A. Piva, "Prnu Pattern Alignment for Images and Videos Based on Scene Content," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019: IEEE, pp. 91-95.

[56] Y. Zhao, N. Zheng, T. Qiao, and M. Xu, "Source camera identification via low dimensional PRNU features," *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 8247-8269, 2019.

[57] P. Yang, R. Ni, Y. Zhao, and W. Zhao, "Source camera identification based on content-adaptive fusion residual networks," *Pattern Recognition Letters*, 2017.

[58] X. Ding, Y. Chen, Z. Tang, and Y. Huang, "Camera identification based on domain knowledge-driven deep multi-task learning," *IEEE Access*, vol. 7, pp. 25878-25890, 2019.

[59] M. H. Al Banna, M. A. Haider, M. J. Al Nahian, M. M. Islam, K. A. Taher, and M. S. Kaiser, "Camera Model Identification using Deep CNN and Transfer Learning Approach," in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 2019: IEEE, pp. 626-630.

[60] X. Kang, Y. Li, Z. Qu, and H. J., "Enhancing source camera identification performance with a camera reference phase sensor pattern noise," *IEEE Transactions on Information Forensics and Security,* vol. 7, no. 2, pp. 393-402, April 2012.

[61] M. S. Behare, A. Bhalchandra, and R. Kumar, "Source Camera Identification using Photo Response Noise Uniformity," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2019: IEEE, pp. 731-734.

[62] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research,* vol. 11, no. 12, pp. 3371-3408, December 2010.

[63] F. Rosenblatt, "Principles of neurodynamics," 1962.

[64] M. Minsky and S. Papert, "Perceptrons," ed: Cambridge, Massachusetts, MIT press, 1969.

[65] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Aistats*, 2010, vol. 9, pp. 249-256.

[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097-1105.

[67] Z. Zheng, Z. Li, and A. Nagar, "Compact deep neural networks for device-based image classification," in *Mobile Cloud Visual Media Computing*: Springer, 2015, pp. 201-217.

[68] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *2014 IEEE International Conference on Image Processing (ICIP)*, October 2014: IEEE, pp. 5132-5136.

[69] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout Networks," *ICML (3),* vol. 28, pp. 1319-1327, 2013.

[70] D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju, "Why Regularized Auto-Encoders learn Sparse Representation?," *arXiv preprint arXiv:1505.05561,* 2015.

[71] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems,* vol. 19, p. 153, 2007.

[72] R. R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in *Proceedings of the 25th international conference on Machine learning*, 2008: ACM, pp. 872-879.

[73] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1974.

[74] G. Tesauro, "Practical issues in temporal difference learning," in *Reinforcement Learning*: Springer, 1992, pp. 33-53.

[75] C. Gravelines, "Deep learning via stacked sparse autoencoders for automated voxel-wise brain parcellation based on functional connectivity," The University of Western Ontario, 2014.

[76] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234-242, March 1992.

[77] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527-1554, May 2006.

[78] M. A. Carreira and G. E. Hinton, "On contrastive divergence learning," in *AISTATS*, 2005, vol. 10: Citeseer, pp. 33-40.

[79] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 735-742.

[80] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1033-1040.

[81] N. L. Roux, P.-A. Manzagol, and Y. Bengio, "Topmoumoute online natural gradient algorithm," in *Advances in neural information processing systems*, 2008, pp. 849-856.

[82] K. Cho, T. Raiko, and A. T. Ihler, "Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 105-112.

[83] H. Jaeger, "Echo state network," *Scholarpedia*, vol. 2, no. 9, p. 2330, 2007.

[84] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Aistats*, 2011, vol. 15, no. 106, p. 275.

[85] Y. Bengio, ., A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798-1828, June 2013.

[86] T. Amaral, L. M. Silva, L. A. Alexandre, C. Kandaswamy, J. M. Santos, and J. M. de Sá, "Using different cost functions to train stacked auto-encoders," in *Artificial Intelligence (MICAI), 2013 12th Mexican International Conference on*, 2013: IEEE, pp. 114-120.

[87] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*, 1st ed. Springer Science & Business Media, 2013.

[88] S. Mannor, D. Peleg, and R. Rubinstein, "The cross entropy method for classification," in *Proceedings of the 22nd international conference on Machine learning*, August 2005: ACM, pp. 561-568.

[89] M. James. "Why you should use cross-entropy error instead of classification error Or mean squared Error for neural network classifier training." https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/ (accessed.

[90] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 265-272.

[91] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in neural information processing systems*, 2008, pp. 161-168.

[92] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical programming*, vol. 127, no. 1, pp. 3-30, October 2011.

[93] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, July 2012.

[94] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th annual international conference on machine learning*, 2009: ACM, pp. 873-880.

[95] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*: Springer, 2012, pp. 437-478.

[96] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, pp. 1-12, September 2016.

[97] Y. B. a. A. C. Ian Goodfellow "Autoencoders." MIT Press. http://www.deeplearningbook.org/contents/autoencoders.html (accessed 24 July, 2018, 2018).

[98] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, November 2014.

[99] M. A. Salama, A. E. Hassanien, and A. A. Fahmy, "Deep belief network for clustering and classification of a continuous data," in *The 10th IEEE International Symposium on Signal Processing and Information Technology,* December 2010: IEEE, pp. 473-477.

[100] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research,* vol. 11, no. Dec, pp. 3371-3408, 2010.

[101] X. Ye, L. Wang, H. Xing, and L. Huang, "Denoising hybrid noises in image with stacked autoencoder," in *Information and Automation, 2015 IEEE International Conference on*, August 2015: IEEE, pp. 2720-2724.

[102] C. Kandaswamy, L. M. Silva, L. A. Alexandre, R. Sousa, J. M. Santos, and J. M. de Sá, "Improving transfer learning accuracy by reusing stacked denoising autoencoders," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014: IEEE, pp. 1380-1387.

[103] T. Gloe and R. Böhme, "The dresden image database for benchmarking digital image forensics," *Journal of Digital Forensic Practice,* vol. 3, no. 2-4, pp. 150-159, 2010.

[104] R. Li, C.-T. Li, and Y. Guan, "Inference of a compact representation of sensor fingerprint for source camera identification," *Pattern Recognition,* vol. 74, pp. 556-567, 2018.

[105] C.-T. Li and R. Satta, "Empirical investigation into the correlation between vignetting effect and the quality of sensor pattern noise," *IET Computer Vision,* vol. 6, no. 6, pp. 560-566, 2012.

[106] S. Timotheou, "A novel weight initialization method for the random neural network," *Neurocomputing,* vol. 73, no. 1, pp. 160-168, December 2009.

[107] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning," in *Proceedings of the 28th international conference on machine learning (ICML-11),* 2011, pp. 1089-1096.

[108] M. Fernandez-Redondo and C. Hernandez-Espinosa, "Weight initialization methods for multilayer feedforward," in *ESANN*, 2001, pp. 119-124.

[109] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.

[110] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568-576.

[111] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*: Springer, 2014, pp. 818-833.

[112] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[113] F.-F. Li, A. Karpathy, and J. Johnson, "CS231n: Convolutional neural networks for visual recognition," *University Lecture*, 2015.

[114] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807-814.

[115] 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.

[116] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580-587.

[117] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," *arXiv preprint arXiv:1501.04587*, 2015.

[118] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[119] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 30, no. 1.

[120] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015)," *arXiv preprint arXiv:1502.01852*.

[121] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[122] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 31-40, January, 2017 2017.

[123] B. Zadrozny, "Reducing multiclass to binary by coupling probability estimates," in *Advances in neural information processing systems*, 2002, pp. 1041-1048.

[124] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers,* vol. 10, no. 3, pp. 61-74, 1999.

[125] W. Chmielnicki and K. Stąpor, "Using the one–versus–rest strategy with samples balancing to improve pairwise coupling classification," *International Journal of Applied Mathematics and Computer Science,* vol. 26, no. 1, pp. 191-201, 2016.

[126] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289-315, August 2007.

[127] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993,* 2016.

[128] M. Dejean-Servières, K. Desnos, K. Abdelouahab, W. Hamidouche, L. Morin, and M. Pelcat, "Study of the impact of standard image compression techniques on performance of image classification with a convolutional neural network," 2017.

[129] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377-2385.

[130] A. Surendranath and D. B. Jayagopi, "Curriculum Learning for Depth Estimation with Deep Convolutional Neural Networks," in *Proceedings of the 2nd Mediterranean Conference on Pattern Recognition and Artificial Intelligence*, 2018: ACM, pp. 95-100.

[131] L. Gui, T. Baltrušaitis, and L.-P. Morency, "Curriculum learning for facial expression recognition," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, 2017: IEEE, pp. 505-511.

[132] T. Yamashita and T. Watasue, "Hand posture recognition based on bottom-up structured deep convolutional neural network with curriculum learning," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014: IEEE, pp. 853-857.

[133] Y. Tang, X. Wang, A. P. Harrison, L. Lu, J. Xiao, and R. M. Summers, "Attention-guided curriculum learning for weakly supervised classification and localization of thoracic diseases on chest radiographs," in *International Workshop on Machine Learning in Medical Imaging,* 2018: Springer, pp. 249-258.

[134] B.-b. Liu, X. Wei, and J. Yan, "Enhancing sensor pattern noise for source camera identification: An empirical evaluation," in *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, 2015: ACM, pp. 85-90.