BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 03/06 pp206–227
Volume 5, Number 3, September 2022
DOI: 10.26599/BDMA.2022.9020006

Deep Feature Learning for Intrinsic Signature Based Camera Discrimination

Chaity Banerjee, Tharun Kumar Doppalapudi, Eduardo Pasiliao Jr., and Tathagata Mukherjee*

Abstract: In this paper we consider the problem of "end-to-end" digital camera identification by considering sequence of images obtained from the cameras. The problem of digital camera identification is harder than the problem of identifying its analog counterpart since the process of analog to digital conversion smooths out the intrinsic noise in the analog signal. However it is known that identifying a digital camera is possible by analyzing the camera's intrinsic sensor artifacts that are introduced into the images/videos during the process of photo/video capture. It is known that such methods are computationally intensive requiring expensive pre-processing steps. In this paper we propose an end-to-end deep feature learning framework for identifying cameras using images obtained from them. We conduct experiments using three custom datasets: the first containing two cameras in an indoor environment where each camera may observe different scenes having no overlapping features, the second containing images from four cameras in an outdoor setting but where each camera observes scenes having overlapping features and the third containing images from two cameras observing the same checkerboard pattern in an indoor setting. Our results show that it is possible to capture the intrinsic hardware signature of the cameras using deep feature representations in an end-to-end framework. These deep feature maps can in turn be used to disambiguate the cameras from each another. Our system is end-to-end, requires no complicated pre-processing steps and the trained model is computationally efficient during testing, paving a way to have near instantaneous decisions for the problem of digital camera identification in production environments. Finally we present comparisons against the current state-of-the-art in digital camera identification which clearly establishes the superiority of the end-to-end solution.

Key words: deep learning; visual signatures; camera identification; convolutional neural networks; deep feature learning

1 Introduction

The availability and use of network connected digital cameras have increased dramatically in the later half of 2020. This growth has been driven by the widespread use of Artificial Intelligence (AI) enabled Internet-of-Things (IoTs), the progress in building self driving vehicles and the easy access to cloud based storage services for storing, sharing and computing with the data generated by these cameras. For example, network connected digital cameras are used in smart surveillance systems for security at homes and connected cities (smart cities)[1,2], smart traffic monitoring systems for real time

• Chaity Banerjee is with Department of Idustrial & Systems Engineering, University of Central Florida, Orlando, FL 32816, USA. E-mail: Chaity.BanerjeeMukherjee@ucf.edu.
• Tharun Kumar Doppalapudi and Tathagata Mukherjee are with the Department of Computer Science, University of Alabama in Huntsville, Huntsville, AL 35806, USA. E-mail: td0057@uah.edu; tathagata.mukherjee@uah.edu.
• Eduardo Pasiliao Jr. is with Air Force Research Labs, United States Air Force, Eglin Air Force Base, Shalimar, FL 32579, USA. E-mail: elpasiliao@gmail.com.
* To whom correspondence should be addressed.
Manuscript received: 2021-11-05; revised: 2022-03-02; accepted: 2022-03-04

tracking of traffic flows, congestion and accidents[3], self driving cars for real time autonomous driving[4], and face recognition based security systems[5] for access control and monitoring of highly secured facilities. Apart from these, there are myriad other application areas where AI enabled systems rely on input from network of cameras for performing specialized tasks.

It must be noted that the AI algorithms learning from multi-sensor data assume that the same is representative of the underlying data generation process (distribution). More precisely, an important assumption of the AI algorithms powering these systems is that the data being generated from the network of sensors (for example a network of digital cameras) are reliable and depict the true state of the world being observed. In other words they assume that the data being obtained from the sensors (or cameras) are not fakes. The advent of deep networks[6] along with the hardware to train and deploy systems using deep architectures, have made it possible to use these structures in applications across different domains. For example, deep networks have been successfully applied to problems in computer vision[7], cyber security[8], computational fluid dynamics[9], medical imaging[10] as well in remote sensing systems[11]. It is no wonder that they have also found successful application in building systems for creating "realistic deep fakes". A "deep fake" of a sensor output is an artificially constructed "realistic looking" fake sensor output obtained using a deep generator architecture, such that it is not possible to distinguish it from the real sensor data[12]. Deep fakes have been successfully constructed for different types of sensor data including voice, video, and image[13]. With the possibility of having deep fakes being used as inputs to an AI inference system, the foundational assumption of the AI algorithms powering these "smart systems" have been demolished. Today we can create realistic looking "deep fakes" that can fool humans into accepting an alternative version of the reality. This in turn poses serious challenges to AI powered systems that rely on data for inference, since now the data can easily be falsified in a way that it is not possible to distinguish between the real and the un-real.

Broadly there are two consequences of using "fake" data with AI systems: when used during training the resulting model will be faulty and will not work well during testing. However when used with models trained on the real data (and hence correct models), it will lead the AI system to make wrong inference during the testing phase. Note that both consequences are the same, namely, faulty inference during testing. Generally, it is harder to trample with the AI algorithm during the training phase due to the rigorous validation that is often times applied during this phase. However, it is easier to attack a trained AI system by feeding it faulty data during testing as it is in general harder to notice and stop such attacks in production. For example, self driving cars use data from a set of sensors in order to make real time driving decisions. An important set of sensors for such tasks is the vision sensors (digital cameras and LiDAR sensors) that capture images, point clouds and videos of the surrounding environment. These are used as inputs to the AI inference engine which makes estimates about the state of the operating environment using this data. Finally these estimates along with estimates from other similar systems are used to make real time driving decisions. It is not hard to imagine what would happen if the data from the cameras are maliciously altered during the operation of a self driving vehicles. Thus one may replace the feed from the front cameras to show that the road ahead is empty when it is actually not, leading to a fatal crash. This happens as the AI engine has no way to recognize that the data it is being fed is fake. This behooves us to build systems for identifying and isolating maliciously altered data before it can be used as input to an AI inference engine.

It must be noted that the general problem of "deep fake" detection is hard and there is no solution that works for all possible scenarios. For example the problem of identifying and isolating maliciously created images can be formulated as follows: given an image I we want to determine whether it was generated by any one of the cameras $\{C_1, \ldots, C_n\}$. Note that in order to solve this problem we need to characterize the joint distribution of the image space of the cameras $\{C_i, i = 1, \ldots, n\}$ as detecting a fake image is equivalent to identifying an item in the complement of the space of the joint distributions. This problem is known to be hard and there is no known algorithm for solving this problem in the general setting. One way of efficiently solving this problem is through the idea of "hardware based camera signatures". Every digital camera has intrinsic hardware characteristics that are superimposed onto the images captured by the camera. If it is possible to "learn" these intrinsic hardware characteristics from the images captured by a digital camera, it would be possible to uniquely identify the given camera. Finally, if this can be done for each of the cameras $\{C_i, i = 1, \ldots, n\}$, then

it would be possible to uniquely characterize the space of all the cameras. Under this setting, given an image I, the system would infer the intrinsic digital camera features from the image and compare it with the known camera features in order to make a determination of the origin of the image. Thus the core of this solution is a system to automatically learn the hardware based intrinsic digital camera features from a given sequence of images. Note that given such a system, it would also be possible to identify a digital camera given an image from the same. Motivated by the aforementioned observation, in this paper we address this problem by designing and implementing an end-to-end system for digital camera identification using deep feature representations from the raw digital camera images.

The use of end-to-end systems for learning hardware based signatures for spoof detection and sensor identification is not new and has been used in other domains before. For example, in the domain of wireless security, the task of radio frequency transmitter identification is important to guard against malicious spoofing attacks. Radio frequency (RF) emitters have intrinsic characteristics that are introduced during the manufacturing process due to imperfections in the fabrication systems. These imperfections introduce unique signatures into the emitted signals that can be learned and exploited for RF emitter identification. For example, Roy et al.[8] designed and implemented an end-to-end system for rogue transmitter detection and emitter identification using hardware signatures learned from raw signal data. Their system could not only identify and isolate malicious emitters, but could also identify known transmitters using their hardware signatures learned with a deep feature learning framework. Digital cameras are nothing but electromagnetic sensors that operate in the visual region of the electromagnetic spectrum (EM spectrum) and hence are similar to RF transceivers albeit the fact that the operating frequencies are different. Consequently, it should be possible to design and build digital camera identification systems in an end-to-end setting using the raw image data. In this paper we take a step in that direction by designing and building such a system and establishing its efficacy using real world custom datasets.

The problem of digital camera identification is harder than the problem of identifying its analog counterpart since the process of analog to digital conversion smooths out the intrinsic noise in the analog signal. However it is known that identifying a digital camera is possible by analyzing the camera's intrinsic sensor artifacts that are introduced into the images/videos during the process of photo/video capture even though such methods are computationally intensive. This problem has been studied for a long time and recently several efficient and robust algorithms for solving this problem have been proposed. It must be noted that most of these solutions have been based on the state-of-the-art algorithm proposed by Lukas et al.[14] The crux of this algorithm is to learn the so-called "sensor pattern noise" from the images using wavelet-based denoising filter. Though this method is efficient, it is computationally intensive and time consuming and hence hard to use in practice.

In this paper we propose an end-to-end system for identifying digital cameras using video sequences obtained from them with a deep feature learning framework. We conduct experiments using three custom datasets: the first containing two cameras in an indoor environment where each camera may observe different scenes, the second containing images from four cameras in an outdoor setting but having the same characteristics as the previous dataset in that each camera may potentially observe different scenes and the third containing images from two cameras observing the same checkerboard pattern in an indoor setting. Our results show that it is possible to learn deep feature representations using an end-to-end system that can capture the intrinsic hardware signature of the cameras, which can then in turn be used to disambiguate the cameras from each another. Moreover our system is end-to-end, requires no complicated pre-processing steps and the trained model is computationally efficient during testing proving a way to have near instantaneous decisions in production environments. Succinctly, our contributions in this paper can be summarized as follows:

(1) Propose, design, and implement a deep learning based end-to-end digital camera identification system using implicitly learned intrinsic features of the cameras obtained from sequence of images obtained from them.

(2) Curate extensive datasets both in indoor and outdoor settings with the goal of testing the efficacy of our algorithm for digital camera identification. The datasets have been collected with different sets of cameras under different settings with the goal of making the identification task realistic.

(3) Establish the efficacy of our proposed end-to-end approach for camera identification through experiments using the curated dataset and comparisons with the state-of-the-art method proposed by Lukas et al.[14]. Precisely,

we show that our method is more accurate and takes less time on the same set of images and the same hardware platform.

The rest of the paper is organized as follows. In Section 2 we present a small survey of the previous work in digital camera identification and then we describe our system in Section 3, the datasets in Section 4 and present the results of our experiments in Section 5. Finally we discuss and conclude our work in Sections 6 and 7, respectively.

2 Previous Work

The problem of identifying a digital camera from image based features has been studied for more than a decade. One of the original motivations for studying this problem was the possibility of its use in forensic investigations where there is need for connecting an image with a specific camera. As noted before, cameras have intrinsic hardware signatures due to manufacturing irregularities in the image capture system which create unique fingerprints in the images obtained from a given camera. It is possible to isolate these features in order to learn an unique signature for a given camera that can be used for identification.

In Ref. [14] the authors presented a digital camera identification system using the idea of *sensor pattern noise*. For each camera they identified and extracted a reference pattern noise that was used as a unique identification fingerprint for the camera. This was done by averaging the noise obtained from multiple images using a denoising filter. Finally, given an image they used a correlation detector to check for the presence of the noise fingerprint as a spread spectrum watermark in the image. The authors presented experimental results with 320 images collected from digital cameras of various makes and models to establish the efficacy of this system. One of the drawbacks of this method is the time required to extract the image reference fingerprint and detect its presence in a test image. As a result, though this method is accurate, its use is limited to scenarios where there are a small number of images and/or cameras or where computation time is not a factor in the choice of the algorithm. Hence this method is not suitable for large scale deployments of digital cameras as is the case today especially with the advent of IoTs and self driving cars.

Similarly in Ref. [15] the authors used the idea of interpolation in the color surface of an image due to the use of a color filter array (CFA) in a digital camera, for creating a blind source camera identification system. They used a set of image characteristics in conjunction with a support vector machine based multi-class classifier to determine the originating digital camera from a given image. In a similar manner, the authors in Ref. [16] proposed a new feature based approach using the idea of photo response non-uniformity (PRNU) noise for the problem of source camera identification. Their method works by choosing the features which are robust for image manipulations. The PRNU noise is extracted from the images using wavelet based denoising method and represented by higher order wavelet statistics which are invariant features for image manipulations and geometric variations. These features are used as input to a support vector machine classifier to identify the source camera for a given image. In Ref. [17] the authors study the robustness of digital camera identification methods based on the use of convolutional neural networks whereas in Ref. [18] the authors proposed a convolutional neural network (CNN) architecture for the problem of source camera identification using images from mobile devices. Though they reported an accuracy of around 98 % their results are limited to images from mobile phones which mostly consist of images of faces, the dataset being part of the Mobile Iris Challenge Dataset[19]. In this work we present an end-to-end system for the task of digital camera identification, similar to the one proposed by Ref. [18], but present experimental results on a more diverse dataset containing images from smart phone cameras and webcams in indoor and outdoor settings. Our accuracy tracks the hardness of the dataset. Finally, our system does not involve any complex pre-processing steps and can scale to a large number of sources without any significant redesigning of the network.

3 Method

We implemented a deep neural network for the task of digital camera identification using raw (or slightly pre-processed) images from a camera as input. Our network takes an image and classifies it as being taken by one of the known cameras that the system was trained to identify. Succinctly, we assume that there are n cameras $\{C_1, C_2, \ldots, C_n\}$ and we have training data from each of these cameras, which are used for training a neural network model $\mathcal{M}$ for the task of camera identification. Given a test image I, the model outputs an index i identifying the camera which was used to obtain this

image.

Intuition: The process of digital image acquisition using a camera superimposes the hardware characteristics of the image acquisition hardware on the acquired image. Succinctly, $I = I_s \oplus I_n$ where $\oplus$ is the convolution operation, it I_s the image signal, and I_n is the noise signal. This convolution in the signal space creates an unique pattern in the image that can be extracted to form a unique signature for the camera. However the signature extracted from a single image will also contain intrinsic properties of the image scene and hence this signature will vary from image to image. In order to smooth out the image specific features, one needs to average out the image centric noise across several images obtained from the camera. Intuitively this "average noise pattern" can be used to uniquely identify a camera. Though this is intuitively appealing, it is computationally intensive to obtain this pattern and use it with images in real time for camera identification. Current state-of-the-art methods use signal processing for extracting the noise pattern and hence can be considered to be expert engineered methods. However, recently deep learning based automatic feature learning techniques have shown promise for a variety of tasks. These methods are capable of learning feature embeddings from the training data and can excel at certain types of learning tasks. Convolutional neural networks are a variation of these architectures that can be used to learn image based features. Since a given image I is a convolution of the image signal I_s and the noise signal I_n, it is natural to expect that convolutional networks will be able to learn noise based features from I through the use of properly designed filters. We expect these features to have enough discriminative information to identify a given camera. Our approach is motivated by this intuition and we use a well known convolutional network for this task. Next we describe the neural network architecture used for this purpose.

Network architecture: We start by noting that the problem of digital camera identification is a classification problem as our goal is to assign an input image to one of the n classes, each class identifying with one of the cameras $\{C_i, i = 1, \ldots, n\}$. Rather than designing our own deep neural architecture for this problem, we decided to use one of the several deep network architectures that are known to work well for the task of image classification. It is known that the accuracy of deep network based classification methods depends on the quality of the feature representations learned by the network. Hence in order to select a network for this task we need to consider the quality of the feature representations being computed by the network. Based on this consideration we decided to use the GoogleNet[20] architecture as it is known to learn good dicriminative features for the task of image based classification[21]. The GoogleNet architecture is shown in Fig. 1 and our modifications to the same is shown

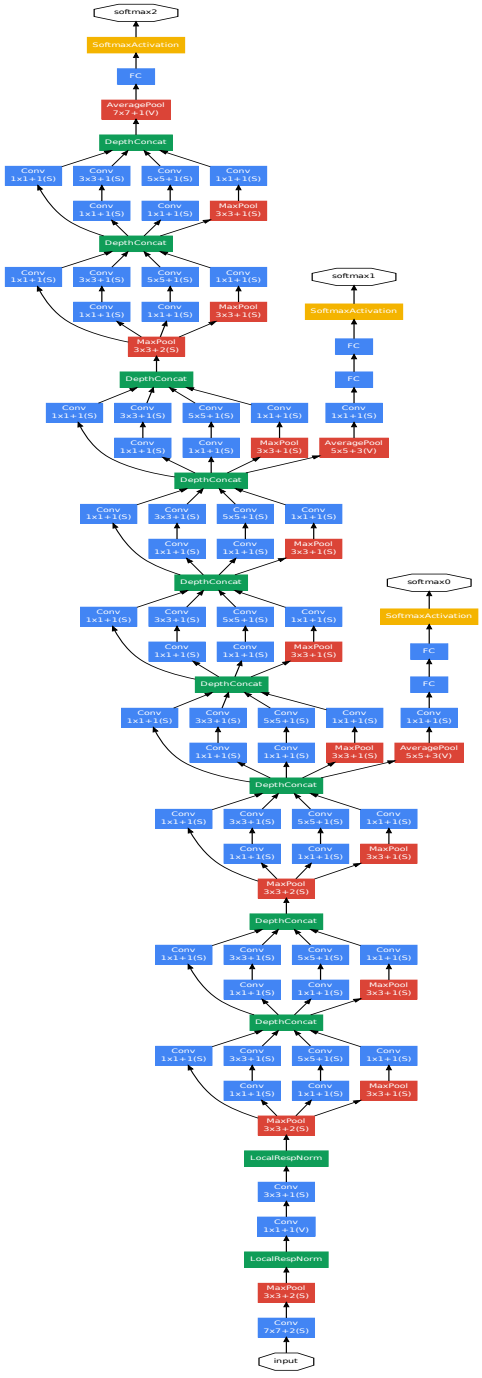


Fig. 1 GoogleNet architecture.

in Fig. 2. Note that GoogleNet contains a total of 27 layers with nine Inception modules and the input to this architecture accepts an image of dimension 224 × 224. We modified this network minimally by adding one extra output layer having n nodes to represent the n output classes corresponding to the cameras and trained the network to identify the camera given an input image.

Training: We trained the network with three different datasets: the first containing two cameras, the second having four cameras, and the final one with two cameras. For the two class classification we used the binary cross entropy loss with the sigmoid activation function in the last layer and for the four class classification we used the sparse categorical cross entropy loss[6] with the softmax activation function in the final layer. We used the ADAM optimizer with a learning rate of $lr = 0.0001$ and initialized the weights of the network with the Glorot initializer[22] before starting the training. Note that GoogleNet uses an input of size 224 × 224 whereas in our case the input images had different sizes as shown in Table 1. As a result we needed to resize the images to 224 × 224 before feeding them into the network.

4 Datasets

For this work we used three custom built data sets: we call the first dataset "the two camera data", the second dataset "the four camera data", and the third dataset "the checkerboard data". Next we describe each of these datasets while explaining their unique characteristics.

4.1 Two-camera dataset

The two-camera dataset was collected in an indoor setting using a robot mounted with four Google Pixel 4 smartphones, two of which were used at any given time to collect the data. A continuous video feed was obtained using the cameras of the Pixel 4 phones, while

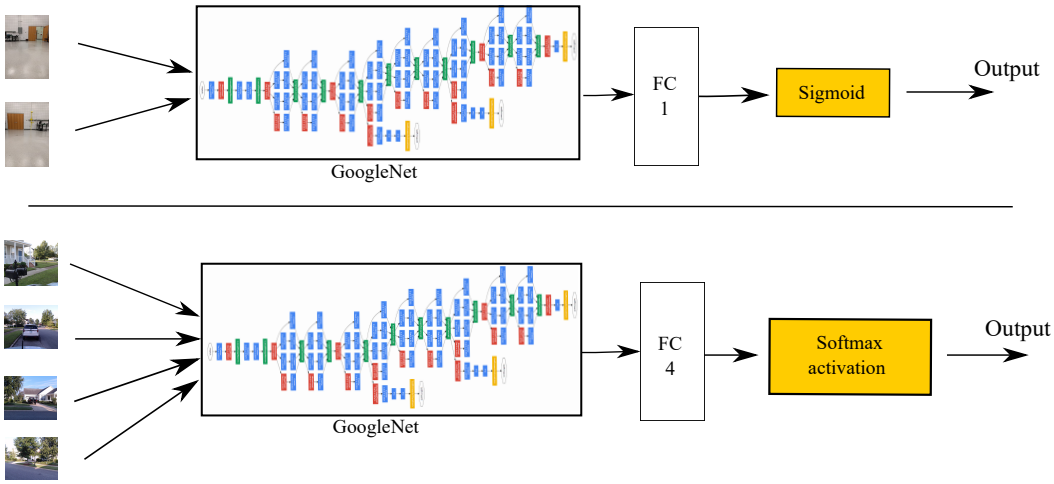


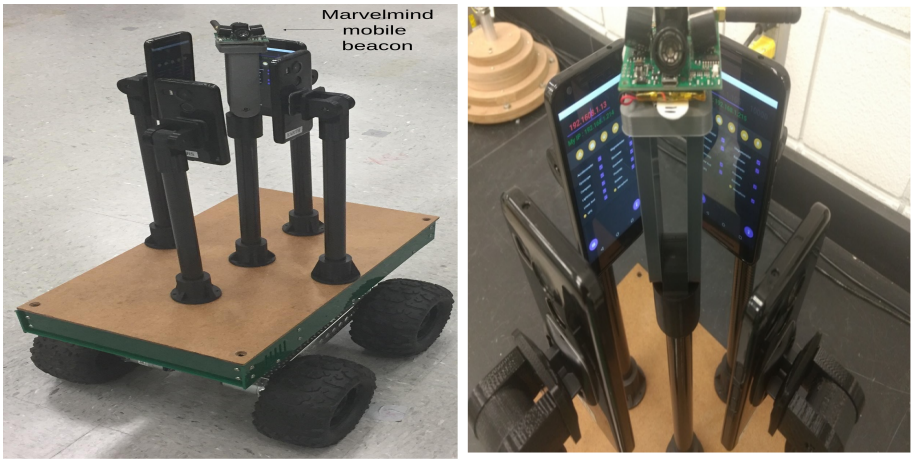
Fig. 2 Camera identification system architecture.

Table 1 Metadata for each of the datasets used for experiments.

Dataset name		Image size	# Images
Two Camera (Indoor)	Recording 1 Front	1080 × 1920	1829
	Recording 1 Back	1080 × 1920	1727
	Recording 2 Left	1080 × 1920	1441
	Recording 2 Right	1080 × 1920	1351
	Recording 3 Left	720 × 1280	13926
	Recording 3 Right	720 × 1280	13 936
	Recording 4 Left	720 × 1280	20 336
	Recording 4 Right	720 × 1280	20 328
Four Camera (Outdoor)	Set 1	640 × 480	1800 (Each Camera)
	Set 2	640 × 480	1800 (Each Camera)
	Set 3	640 × 480	1800 (Each Camera)
	Set 4 (middle 1 min)	640 × 480	1800 (Each Camera)
Checkerboard (Indoor)	Set 1 (Identical)	1920 × 1080	2403 (Each Camera)
	Set 2 Camera 1	1920 × 1080	2403 (Each Camera)
	Set 2 Camera 2	1280 × 720	2403 (Each Camera)

driving the robot around using hand held controllers along the corridors of the building and inside a laboratory room. The robot was fitted with an ultrasound based indoor positioning system for precise control and an Intel Nuc computer running Ubuntu 16.04 for general purpose computing and networking needs. The setup and a closeup of the same are shown in Fig. 3. We used a custom built application which was installed on the phones for interacting with the cameras as well as with the Nuc. The application was responsible for obtaining the video feeds and passing it onto the Nuc over a personal hotspot running off the phone, to which the Nuc was connected. We can see part of the interface of the phone app in Fig. 3. Figures 4 and 5 show sample images from the back, front, left and right cameras, in the laboratory room and the corridor respectively. It must be noted that the front and back cameras, in the examples shown, see images that have considerably different set of features whereas the left and right cameras see images where the features are mostly uniform. Intuitively, it would be harder to distinguish between the cameras if the features obtained from them are similar since in this situation there would not be many discriminating artifacts to separate one camera from the other in the feature space. This dataset was collected over a four day period, each day being in one part of the building. As a result this dataset contains four recordings where each recording consists of a continuous sequence of video feed obtained from a pair of cameras.

Processing: In order to process the video feed from each of the cameras, we used OpenCV[23] for extracting the image frames from each of the videos. A second of video feed resulted in approximately 30 image frames being extracted and we extracted all the frames from the entire video feed for creating the dataset corresponding to each of the cameras. Note that for some of the videos,



(a) (b)

Fig. 3 (a) Experimental setup where four cameras are mounted on the ground robot; each camera is perpendicular to the other. The indoor positioning receiver is at the center for accurate control of the robot. (b) Closeup of the camera setup.

Fig. 4 Sample image from back & front camera in the laboratory room.

Fig. 5 Sample image from left & right camera in the laboratory corridor.

each second of video feed gave us either less than or more than 30 frames and hence the total number of images in the dataset corresponding to each of the cameras is not a multiple of 30. The details of the metadata of this dataset is provided in Table 1.

4.2 Four-camera dataset

The four-camera dataset was collected in an outdoor environment in a city in the state of Florida. We built a custom camera mount system that can be mounted on the roof of a car. The mount contains eight Logitech webcams arranged in a circle such that the field-of-view (FoV) of each of the cameras do not intersect considerably with the adjacent ones (both on the right and left of each camera). Note that the adjacent cameras have a small intersection in their FoV and we decided to use only four of them at any given time for data capture. The cameras are selected in a way that they observe scenes that have image feature overlaps making it hard to disambiguate between them purely based on the image based features. Precisely, we selected the cameras in the positions shown in Fig. 6 while the rest of them were turned off. The cameras are connected to two Intel Nuc computers running Ubuntu 16.04 and powered by a lithium ion battery pack. The setup was mounted on the roof of a 2012 Ford Focus and driven around at posted speed limits (varying between 40–56 km/h), while obtaining video feed from the cameras. The overview of the camera mount system is shown in Fig. 7 while the details of the mounting system is shown in Figs. 8 and 9.

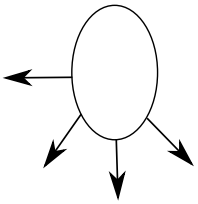


Fig. 6 Position of selected cameras for data collection.

(a) (b)

Fig. 7 (a) The car camera mount system. There are eight cameras in total out of which only four were used for data collection. (b) Bottom suction setup for mounting on top of car.

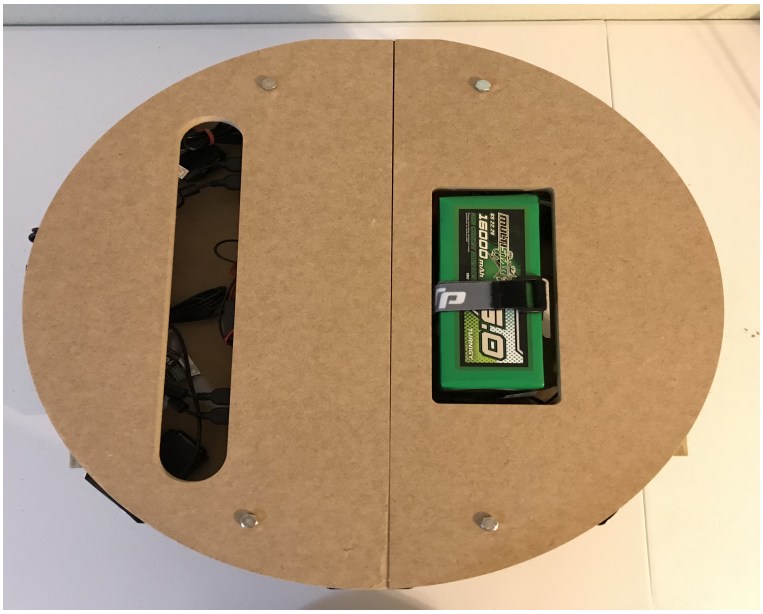


Fig. 8 Top view of camera mount system.

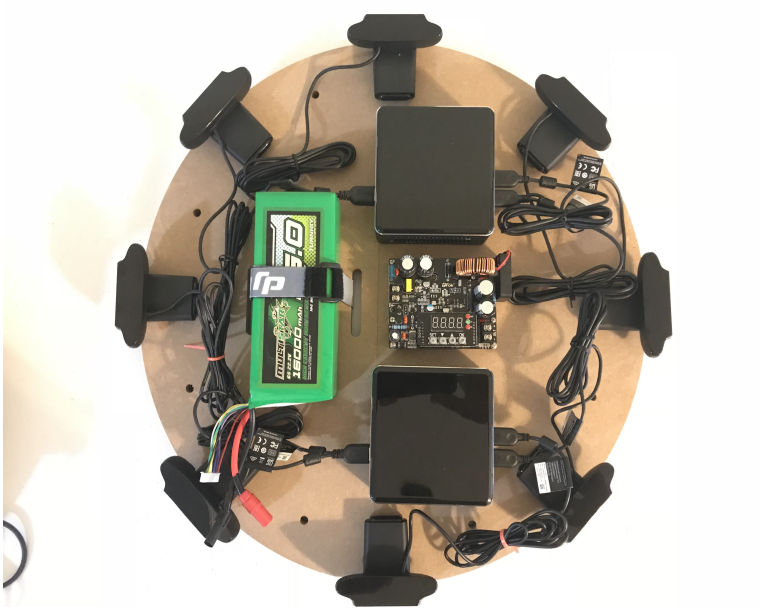


Fig. 9 Inside view of camera mount system. There are two Intel Nucs that record the video feeds from the cameras.

An example of the images collected by the four camera setup is shown in Figs. 10 and 11, where Fig. 10 shows the images from cameras 1 & 2 and Fig. 11 shows the images from cameras 3 & 4. It must be noted that the images see disjoint parts of the scene with no overlap in the images. Finally we also note that the images in this case are more diverse compared to the two camera images as these are captured in an outdoor setting. It must also be pointed out that since the car was driven around a locality in the city, the same scene might have been observed by different cameras at different times, this making the task of camera identification harder.

Pre-processing: In this dataset there were four sets of video feeds, one from each of the cameras. The raw video feeds were approximately an hour long each and hence in order to process the data within the constraints of the time and resources that we had, we split each of the video feeds into three sets. Set 1 and the Set 3 were for one minute each whereas the Set 2 was for four minutes. The Set 1 contains the recording for the first one minute of the video from each of the cameras and the Set 3 contains the same but for the last minute of the

Fig. 10 Sample images collected using the cameras. Left: camera 1 & right: camera 2.

Fig. 11 Sample images collected using the cameras. Left: camera 3 & right: camera 4.

video whereas the Set 2 contains four minutes of video feed from each of the cameras obtained at a random location in the middle of the video. The details of the metadata of this dataset are provided in Table 1. Note that we also used Set 4 from a random one minute clip from the middle of the videos. However since the car was moving at a speed of 35 km/h, this one minute clip did not have enough changes in the features seen by the cameras in order to learn enough discriminative features and hence the network overfit by a small amount. In order to correct this we selected a longer period of time and hence the four minute clips. We discuss this in detail in Section 6.

4.3 Checkerboard dataset

The checkerboard dataset was collected in an indoor laboratory setting. It was collected with two cameras and contains two different datasets. The first dataset was collected using cameras from different manufacturers whereas the second one was collected using two cameras from the same manufacturers. In the first case we used a Logitech webcam and a Adesso Cybertrack webcam while in the second case we used two Adesso Cybertrack webcams. In each case we collected a video sequence using the webcams in a setting where the webcams were focused in a checkerboard pattern printed on a cardboard box. This was done in order to ensure that each camera observes the same set of features, thus making the task of camera identification harder. Furthermore using cameras from the same manufacturer allows us to test the robustness of our system, when each camera observes similar patterns, thus making discriminative features hard to learn. The sample images are shown in Figs. 12 and 13. Note that the cameras capture images of the checkerboard pattern along with other artifacts in the room. Since the data was collected at the same time, both the cameras see the same items in the same relative position with each other and with respect to the camera.

Processing: In order to process the video feed from each of the cameras, we used OpenCV[23] for extracting

Fig. 12 Sample images collected using same cameras. Left: camera 1 & right: camera 2.

Fig. 13 Sample images collected using different cameras. Left: camera 1 & right: camera 2.

the image frames from each of the videos. Each video was for a duration of 1 minute and 20 seconds. A second of video feed resulted in approximately 30 image frames being extracted and we extracted all the frames from the entire video feed for creating the dataset corresponding to each of the cameras. Note that for some of the videos, each second of video feed gave us either less than or more than 30 frames and hence the total number of images in the dataset corresponding to each of the cameras is not a multiple of 30. The details of the metadata of this dataset is provided in Table 1.

5 Experiments & Results

For each dataset we used 80% of the data, selected randomly, for training and the remaining 20% of the data for testing. All the experiments were conducted on a system having an AMD Ryzen thread-ripper processor, 64 GB of RAM, a NVIDIA 2080 Ti GPU running Ubuntu Linux at version 20.04. We used a batch size of 25 and trained the network for 300 iterations. Note that we did not use an early stopping criteria and let the training proceed till the end. We used the training set to train the network architecture described in Section 3 and tested the trained network on the randomly selected test data. The experiments were repeated for each of the datasets and its subsets as described in Table 1. We report the training and test accuracy for each of the datasets in Table 2. It must be pointed out that the accuracy as reported in the tables does not give a complete picture of the performance of the system. A more intuitive method is to present the confusion matrix that shows the true positive, false positive, false negative, and true negative in the form of a matrix for each of the classes under consideration. This presents an insight into how the system works by identifying the correct classes of the input and also shows how many of the input values are wrongly classified and how. Since this is a standard technique in the data science community,

Table 2 Training and test accuracy for different datasets. (%)

Dataset name		Training accuracy	Test accuracy
Two Camera (Indoor)	Recording 1	99.78	99.71
	Recording 2	100.00	100.00
	Recording 3	92.88	92.26
	Recording 4	98.24	98.20
Four Camera (Outdoor)	Set 1	97.62	97.22
	Set 2	97.04	96.98
	Set 3	98.62	98.26
	Set 4 (middle 1 min)	97.06	95.69
Checkerboard (Indoor)	Set 1 (Identical)	92.74	91.58
	Set 2	100.00	100.00

we present the confusion matrices of the two-camera dataset (recordings 1, 2 and 3, 4), the four-camera data and the checkerboard data in Figs. 14–18, respectively.

6 Discussion

In this section we discuss some of the observations from our experiments. Specifically we look at the features learned through the first convolutional layer and finally the features learned by the entire trained network. Finally we revisit the accuracies from Table 2 and try to explain the observations in light of the feature maps learned by the network for each of the datasets.

6.1 Features from convolution

We start by looking at the features learned by the first convolutional layer of the network. Figure 19 shows an image from the first recording of the two camera system and the corresponding feature map learned from the first convolutional layer. We observe that the features not only accurately capture important artifacts from the scene, but it also captures the noise in the image as introduced by the camera used for capturing the same. This is important as the image features combined with the noise can create unique signatures that can in turn be used for identification of the source camera. The features learned in this layer can be used by the subsequent hidden layers to synthesize these unique camera signatures. We see the same pattern for the

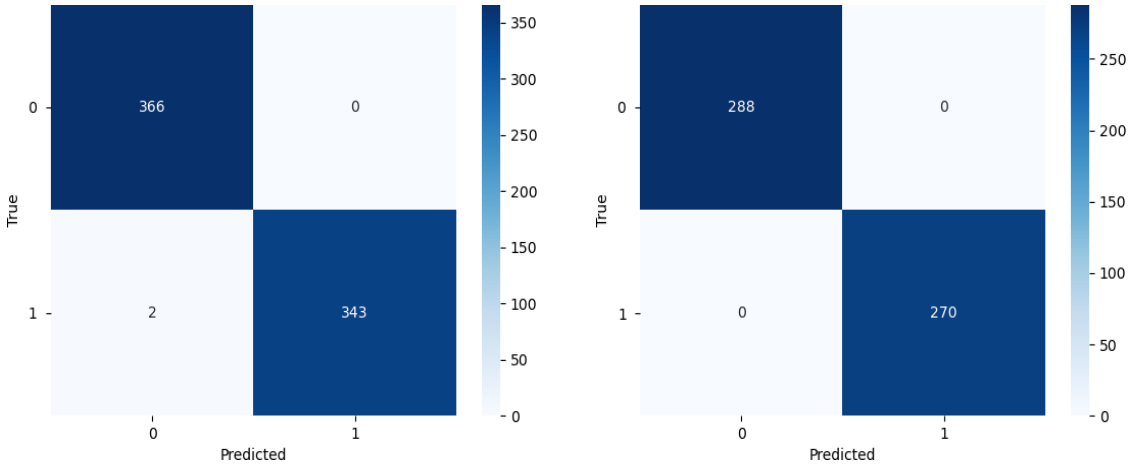


Fig. 14 Confusion matrices for the two-camera dataset: Left: Recording 1, Right: Recording 2.

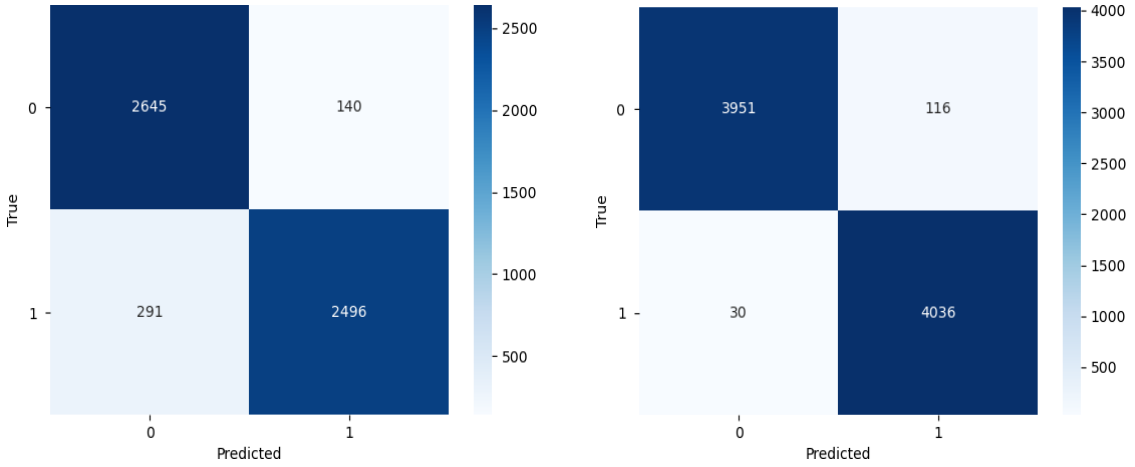


Fig. 15 Confusion matrices for the two-camera dataset: Left: Recording 3, Right: Recording 4.

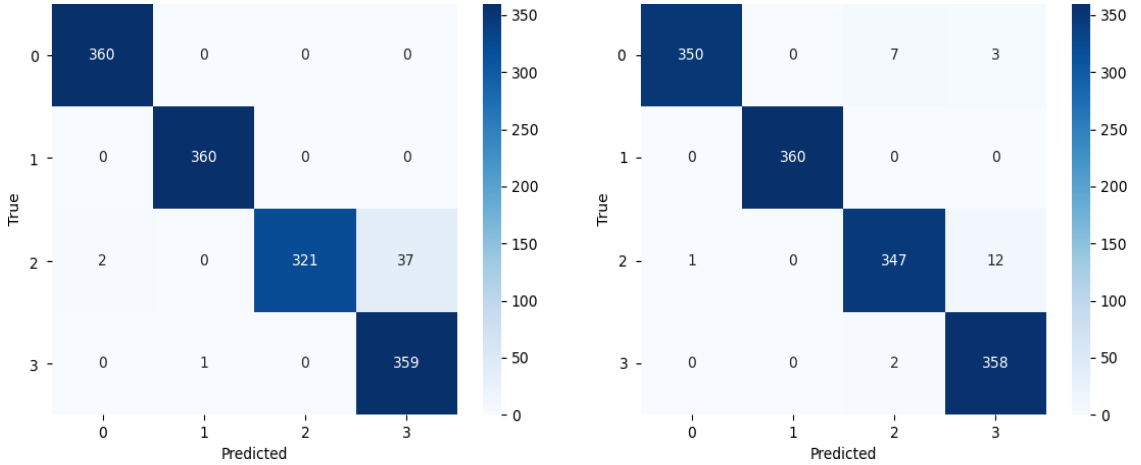


Fig. 16 Confusion matrices for the four-camera dataset: Left: First one minute recording, Right: Last one minute recording.

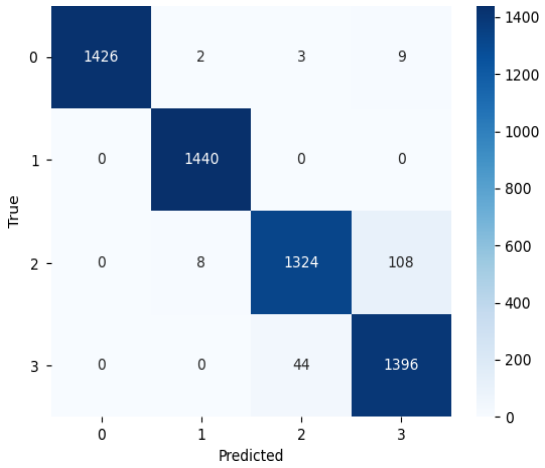


Fig. 17 Confusion matrix for the four-camera dataset: middle four minute recording.

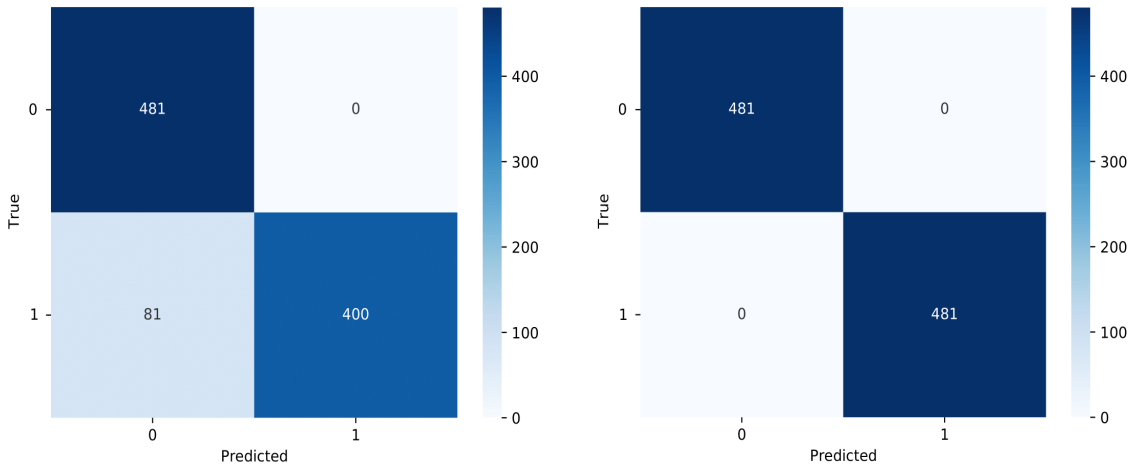


Fig. 18 Confusion matrices for the checkerboard camera dataset: Left: Identical camera manufacturer, Right: Different camera manufacturer.

images from recording 4 of the two-camera dataset (Fig. 20) and a sample image from the four-camera dataset (Fig. 21). We have found the same pattern for all the images from other datasets but we refrain from showing them here due to space constraints.

In order to understand why the network is able to learn features that are able to discriminate between different cameras, we looked at the checkerboard dataset where the two cameras were looking at the same checkerboard pattern. We looked at the feature maps learned from

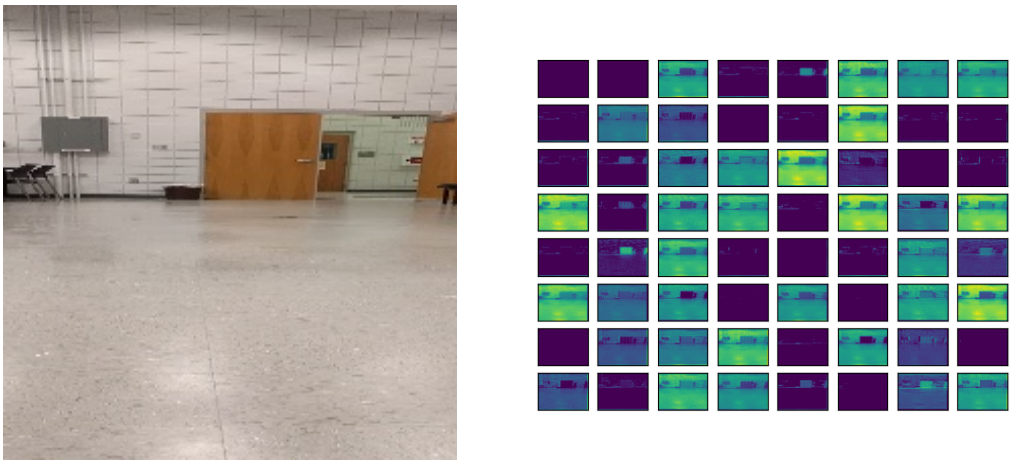


Fig. 19 Sample image from two-camera dataset recording 1 and the corresponding feature map learned by the first convolutional layer.

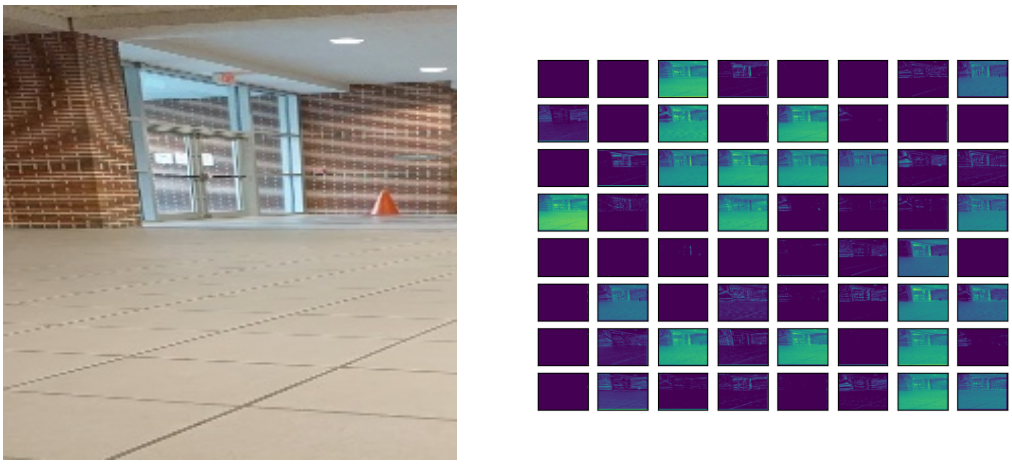


Fig. 20 Sample image from two-camera dataset recording 4 and the corresponding feature map learned by the first convolutional layer.

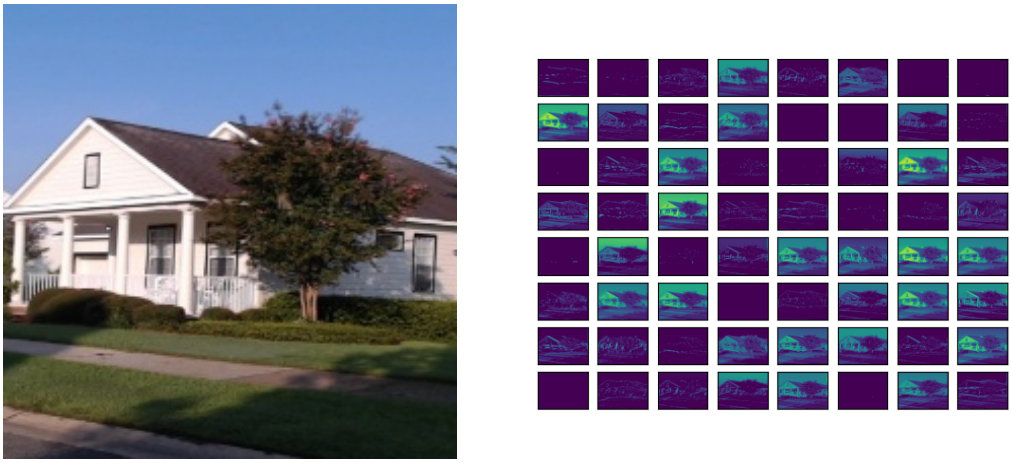


Fig. 21 Sample image from four-camera dataset Set 1 of recording and the corresponding feature map learned by the first convolutional layer.

the first convolutional layer for one image from the first camera and the corresponding image from the second camera. The images and the corresponding feature maps are shown in Figs. 22 and 23. Note that these are the images of approximately the same scene as seen by two cameras from different manufacturers.

Fig. 22 Corresponding images from cameras 1 and 2 of the checkerboard dataset where the cameras are from different manufacturers.

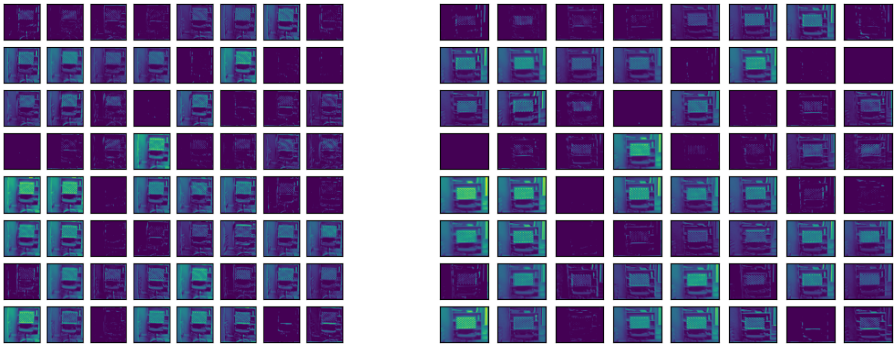
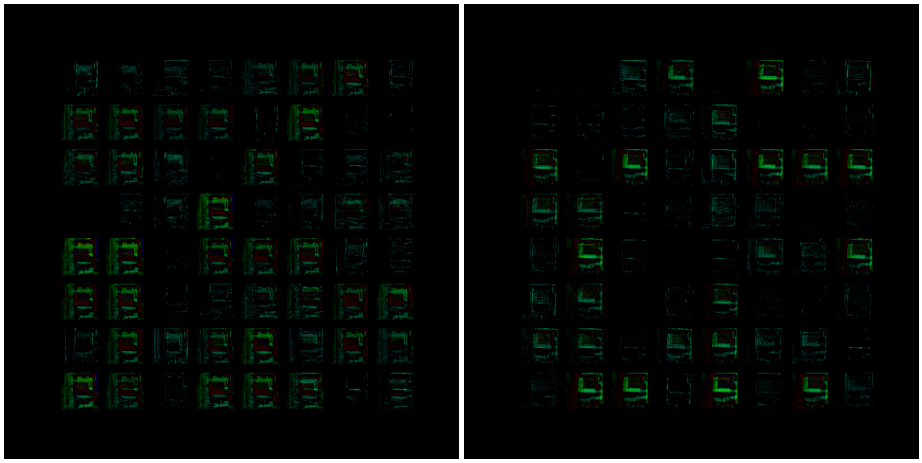


Fig. 23 Feature maps learned by the network after first convolutional layer for images shown in Fig. 22.

Now we look at the difference between the learned feature maps. The intuition is that if the feature maps do not contain any discriminative information, then the difference image would be mostly featureless. However, if there are differences in features between the two cameras, then the difference image would be able to capture these differences. The difference image for the feature maps from Fig. 23, which shows images from cameras having different manufacturers observing an approximately similar scene, is shown in Fig. 24. We observe that the difference image of the feature maps is non-trivial and contains features that capture the differences in the feature maps of the corresponding images. Intuitively, the features contributing to the difference map are being learned by the network from each image and finally mapped into a higher dimensional



(a) (b)

Fig. 24 (a) Difference of feature maps shown in Fig. 23, (b) Same image for identical cameras.

feature space in a way that the images from two different camera, even though of approximately identical scenes, are mapped to different regions in the high dimensional feature space. This in turn allows the network to discriminate between the cameras, based on the image features. The difference image for the case of two identical cameras looking at an approximately similar scene is shown in Fig. 24b. We see a similar type of difference image as in the previous case, that captures difference in the signatures between the two cameras. This is enhanced by the network to learn discriminative features in the high dimensional space. The fact that the features learned in the high dimensional space are separable according to the source cameras, is clearly established by the plots of the principle components and the t-SNE of the features learned in the penultimate layer of our network. We look at these next.

6.2 Final feature maps & accuracy

Here we look at the final feature maps learned by the network. Note that the final features learned by the network is computed by the penultimate layer of the network, which in our case has a dimension of 2048. As a result, the latent feature space learned by the network has dimension 2048. In order to visualize the features in this layer we use two methods, namely, the principal components analysis (PCA) and t-SNE[24,25]. The results are shown in Figs. 25–34. We observe that in almost all the cases the features corresponding to each camera are separated out in well defined clusters. Of particular interest is the case of the checkerboard dataset when the cameras are from the same manufacturer. From Table 2 we observe that the test accuracy for this case is 91.58% which is considerably lower than that of the case of two cameras from different manufacturers. The training accuracy for this case is also lower than the corresponding case of cameras from different manufacturers. We can partially explain this observation from the PCA and t-SNE plots of Fig. 33. We see that there is a considerable overlap between the features learned from the two cameras. Thus, even though for the most part the features are well separated, there is a region in the feature space where they overlap and this

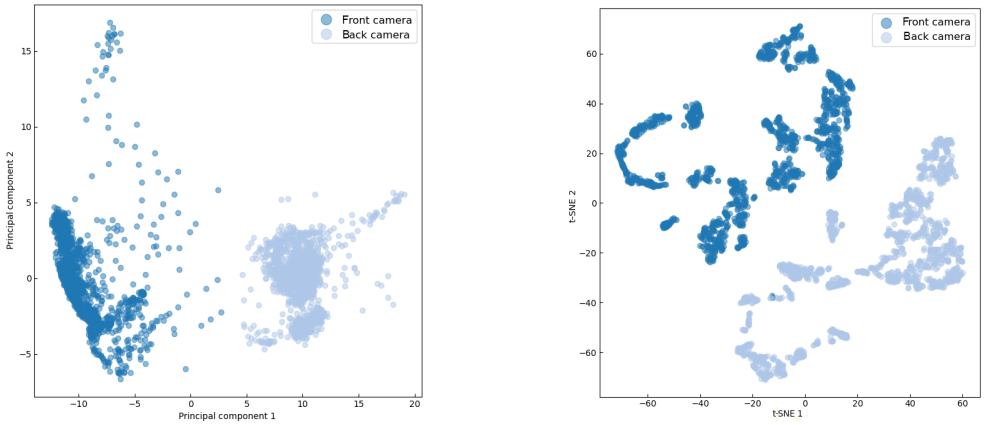


Fig. 25 **Cluster of projection of final feature maps learned by the network for recording 1 of the two-camera dataset.**

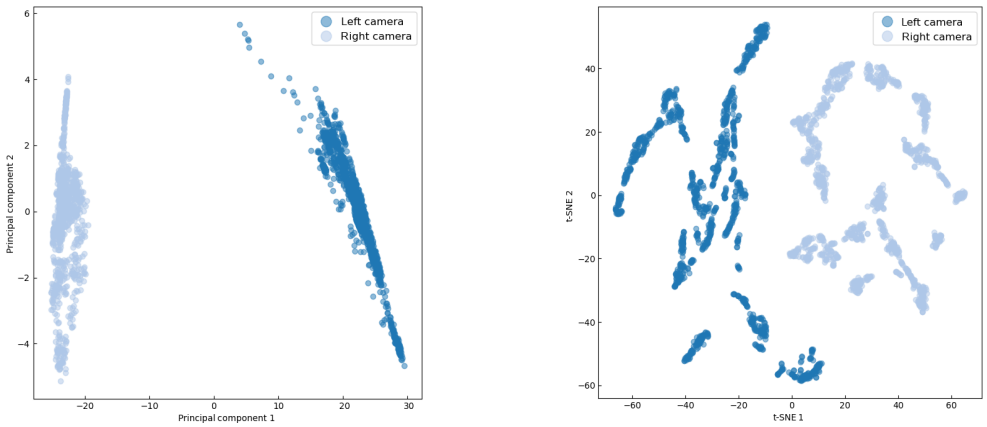


Fig. 26 **Cluster of projection of final feature maps learned by the network for recording 2 of the two-camera datasets.**

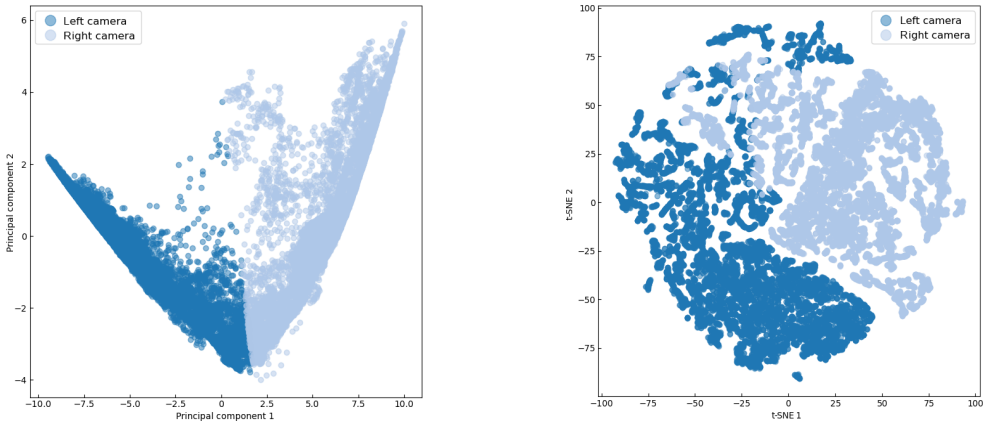


Fig. 27 Cluster of projection of final feature maps learned by the network for recording 3 of the two-camera datasets.

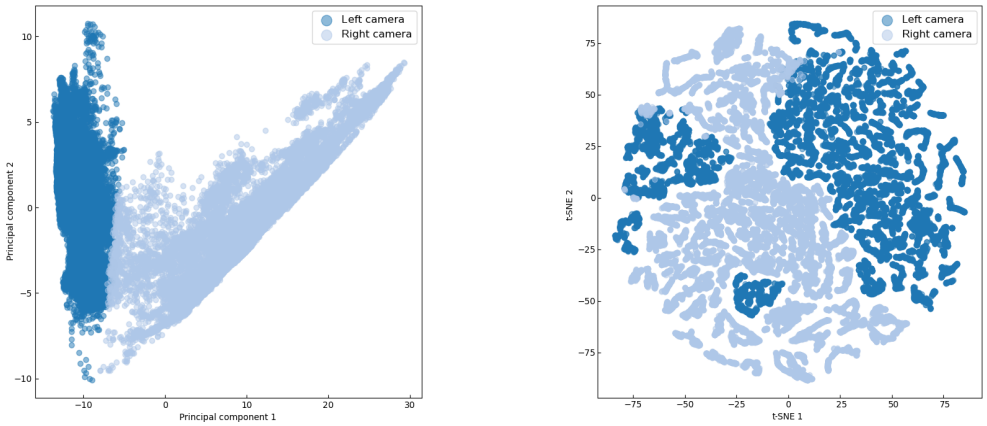


Fig. 28 Cluster of projection of final feature maps learned by the network for recording 4 of the two-camera datasets.

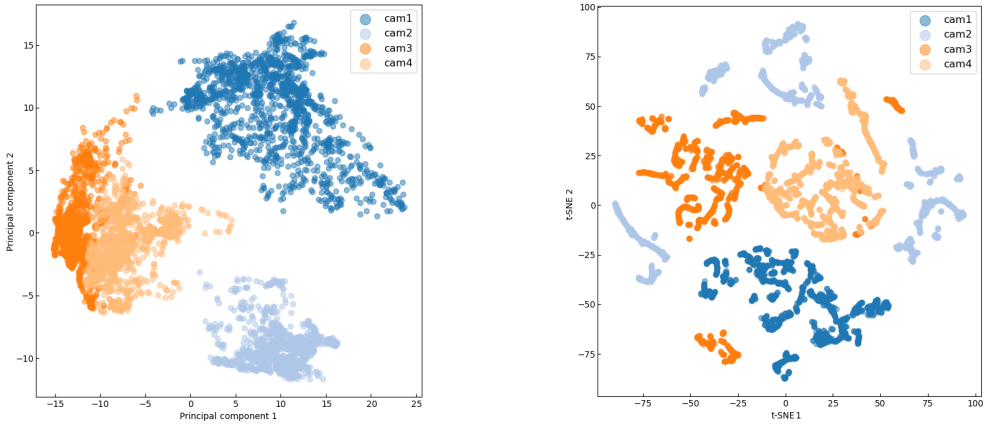


Fig. 29 Cluster of projection of final feature maps learned by the network for Set 1 in the four-camera dataset.

intersection results in difficulty of categorizing the input as being from one camera or the other. We also observe from Fig. 34 that for the case of cameras from different manufacturers, the features learned by the system are well separated into distinct clusters, thus allowing for accurate disambiguation.

Note that we can draw similar conclusions for the case of the third recording from the two-camera dataset. As seen from Fig. 27, it is evident that the features learned by the network overlap in the feature space resulting in the difficulty of disambiguation.

6.3 Time complexity

In this section we present the time complexity of our

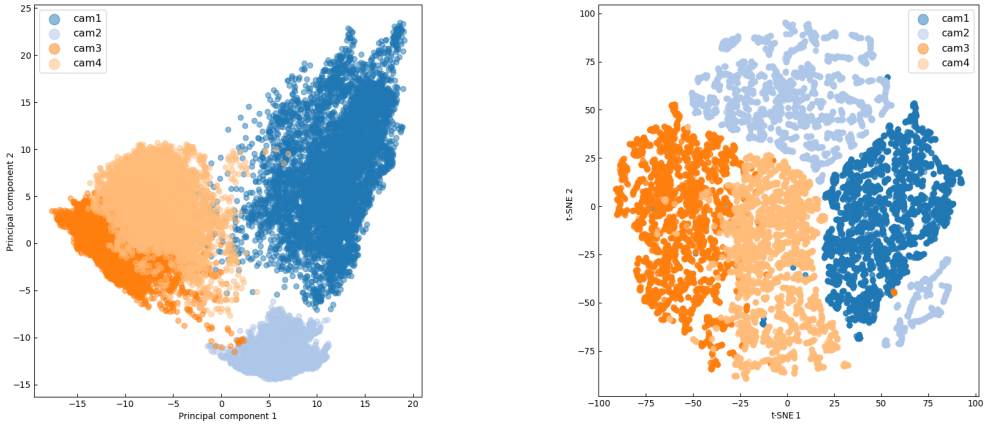


Fig. 30 **Cluster of projection of final feature maps learned by the network for Set 2 in the four-camera dataset.**

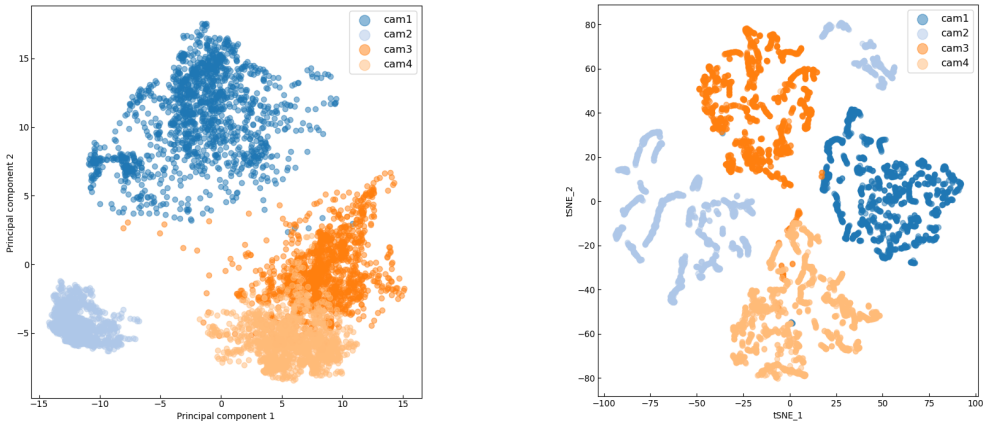


Fig. 31 **Cluster of projection of final feature maps learned by the network for Set 3 in the four-camera dataset.**

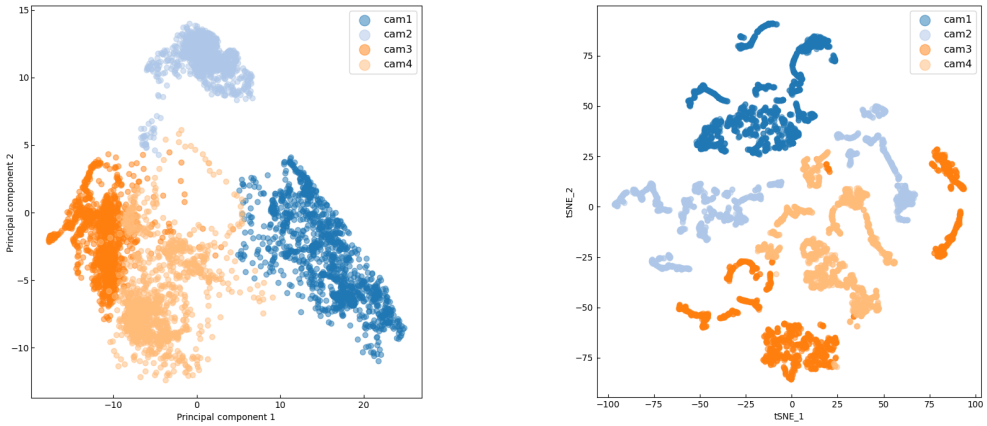


Fig. 32 **Cluster of projection of final feature maps learned by the network for Set 4 in the four-camera dataset.**

model for the task of digital camera identification. It must be pointed out that training the network can be considered as a pre-processing step and hence the training time can be considered as a overhead that does not add to the actual computation time of the network during the testing phase. Moreover, in theory the testing phase requires $O(1)$ time and hence can be considered to be a constant time operation per input. However, in practice this time might vary depending on the neural network architecture and the implementation. Hence it is important to consider the empirical running time for the testing phase in order to understand the efficacy of the network during deployment. As a result when we talk about the time complexity of a neural network, it is

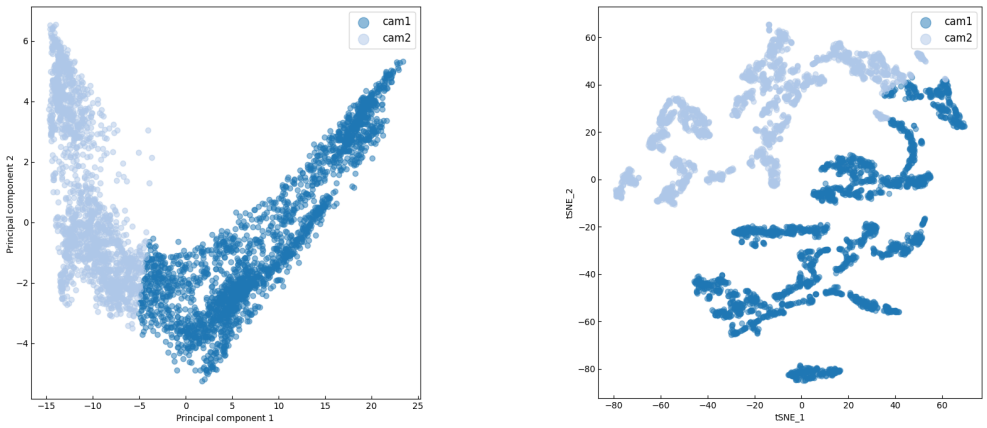


Fig. 33 Cluster of projection of final feature maps learned by the network for the checkerboard dataset with the same cameras.

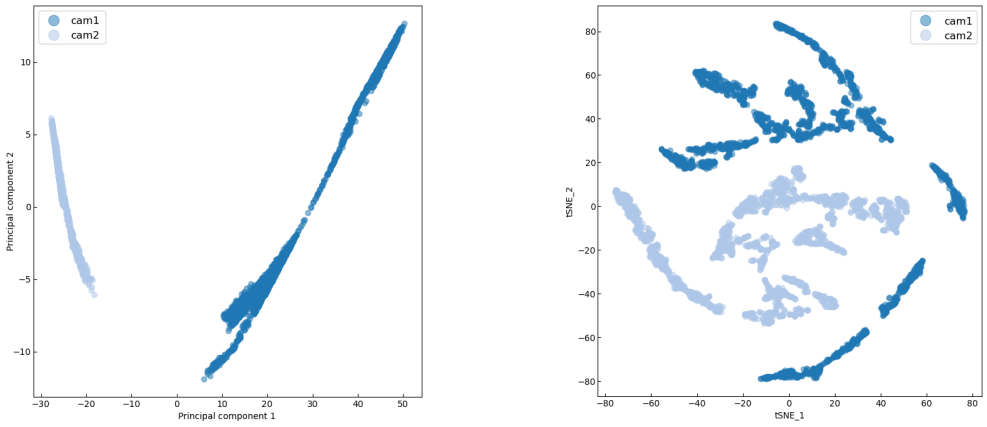


Fig. 34 Cluster of projection of final feature maps learned by the network for the checkerboard dataset with two different cameras.

usually the average testing time of the network computed empirically. For the training time, people are usually interested in theoretical bounds on the time to train a given network architecture. Here we first present the empirical results on the running time of our network during testing with each dataset.

Table 3 shows the average testing time of our trained neural network model on 100 images for each of the datasets. Note that these times are computed with our reference implementation without sophisticated code optimizations that are typically employed in production systems. Based on these numbers we can conclude that the average testing time for an image using this model is approximately 0.1 s. We must point out that for most applications of digital camera identification, this is an acceptable ballpark time. Moreover as shown in Section 6.4, the current state of the art for this task using noise based correlation, requires considerably larger time on the same datasets.

Table 3 Average testing time with 100 images from each dataset.

Dataset name		Average time (s)
Two Camera (Indoor)	Recording 1	14.488
	Recording 2	14.387
	Recording 3	14.376
	Recording 4	14.386
Four Camera (Outdoor)	Set 1	14.753
	Set 2	14.727
	Set 3	14.873
	Set 4 (middle 1 min)	14.933
Checkerboard (Indoor)	Set 1 (Identical)	15.033
	Set 2	14.757

Understanding the time complexity of training a neural network is still an evolving research area. In Ref. [26], the authors proved that a neural network of depth δ can be learned in $poly(s^{2^{\delta}})$ time, where s is the dimension of the input, and $poly(\cdot)$ takes a constant time depending on the configuration of the system. However, the convolution operations of CNN add additional time complexity along with the forward and back-propagation operations. In Ref. [27], the authors mentioned that the

time complexity for training all the convolutional layers is: $O(\sum_{\tau=1}^{\xi}(\eta_{\tau-1}\nu_\tau^2 \cdot \eta_\tau\rho_\tau^2)$, where ξ is the number of convolutional layers, τ is the index of a convolutional layer, $\eta_{\tau-1}$ is the number of input channels of the τ-th layer, ν_τ is the spatial size of the filters at the τ-th layer, η_τ is the number of filters at the τ-th layer and ρ_τ is the size of the output features of the τ-th layer. Using these results one can compute a theoretical bound on the training time of our network. We refrain from getting the exact bound here as it is only of theoretical interest and we see no meaningful purpose being served by this effort. Interested readers can substitute the values of the associated parameters from the neural network architecture and get the bound if desired.

6.4 Comparison with state-of-the-art

The current state-of-the-art for digital camera identification is based on work by Lukas et al.[14] As discussed in Section 2, this work uses the idea of "sensor pattern noise" if an image for identifying the camera that was used to obtain the image. Given reference images from a camera, the algorithm first computes the "reference pattern noise" for the same. Note that this is equivalent to training our neural network. Finally, given an image the algorithm extracts the "sensor pattern noise" from the image and computes the correlation between this noise and the "reference pattern noise" for all the candidate cameras. Then it uses a threshold based method to identify the camera. We used the reference implementation available from https://github.com/andrewlewis/camera-id for this work. Here we present the results of using this method on the second recording from the two-camera dataset and the identical camera data from the checkerboard dataset (see Table 4). We contrast these with the results obtained from our method. Note that we have tried using this algorithm with data from the four-camera dataset but the algorithm was not able to successfully learn the sensor pattern noise from the data. This happened with all the recordings from the four-camera dataset. This might be due to the fact that the cameras being used for obtaining the four camera data produce high resolution images and hence the algorithm fails to extract the sensor noise pattern. Though this is something that we did not expect, but this also goes on to show one of the advantages of our approach. Moreover note that for the second recording of the two-camera dataset, the time required to obtain the "reference pattern noise" is around 4 hours. This is in contrast to the time required for training our system on the same dataset which is approximately 30 minutes using the same hardware. All of the recordings in the four camera data is larger than the two camera one and hence if Lukas et al. algorithm[14] would have worked, it would have required more training time with this dataset. This is also one of the disadvantages of this method.

Table 4 Average testing time with 100 images from each dataset for Lukas method.

Dataset name		Average time (s)
Two camera (Indoor)	Recording 2	605.834
Checkerboard (Indoor)	Set 1 (Identical)	606.74

For the second recording of the two-camera dataset, our algorithm has zero errors. However for the same dataset the algorithm of Lukas et al. makes a total of 206 mistakes evenly spread across both the classes (see Fig. 35). Thus for example, 101 instances from camera 1 are mis-classified as coming from camera 2 whereas 105 instances from camera 2 are classified as camera 1. This clearly shows that our method outperforms the

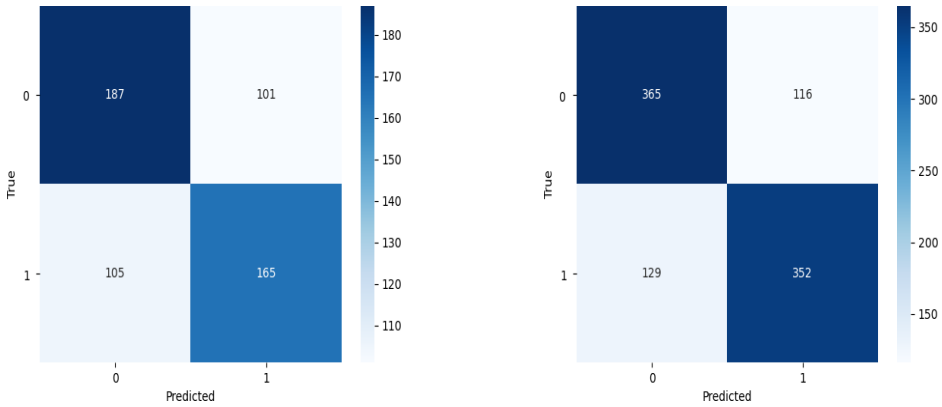


Fig. 35 Results from Lukas method. Left: Confusion matrix for recording 2 of the two-camera dataset. Right: Confusion matrix for checkerboard dataset with identical cameras.

state-of-the-art on this dataset. Finally, we would also like to point out that the our method also gets the best of Lukas et al.[14] in terms of the empirical running time both at the training and test phases.

For the checkerboard dataset with identical cameras (see Fig. 35), our algorithm makes 81 mistakes all for the second camera images which are predicted as being taken by the first camera. However using Lukas et al. algorithm, the total number of mistakes is 245 split over both the classes almost evenly. Thus out of the total of 245 instances, 116 instances of camera 1 are classified as coming from camera 2 whereas 129 instances of camera 2 are classified as coming from camera 1. This clearly establishes the efficacy of our algorithm over the method of Lukas et al. We must point out that the checkerboard dataset with two cameras is the hardest dataset among all the datasets. Even then our algorithm beats the algorithm of Lukas et al. by almost a factor of three in terms of accuracy. In terms of empirical running time during the training and test phases, our algorithm also gets the best of the state-of-the-art.

7 Conclusion

In this work we have shown an implementation of an end-to-end digital camera identification system using images from the cameras. We have conducted extensive experiments with data collected in both indoor and outdoor settings, using inexpensive cameras and have demonstrated the possibility of using such systems for building automated computationally efficient digital camera identification systems. Our work builds upon the success of deep neural networks for image processing albeit the fact that the system is intuitively processing the difference of the signatures of the cameras as learned through the convolutional layers and enhancing the same to learn discriminative features in a high dimensional feature space. Going forward we would like to explore more on the "explainability" aspect of this system. More precisely, we would address the question of why the features learned by the network result in accurate discrimination between the cameras. We would also want to understand the intrinsic nature of the learned features and map them back to specific artifacts of the images. That would be the final objective and would help us not only understand this system but the learning process of neural networks in general.

References

[1] P. Rai and M. Rehman, ESP32 based smart surveillance system, in *Proc. 2019 2nd Int. Conf. Computing, Mathematics and Engineering Technologies*, Sukkur, Pakistan, 2019, pp. 1–3.

[2] M. A. Alsmirat, Y. Jararweh, I. Obaidat, and B. B. Gupta, Internet of surveillance: A cloud supported large-scale wireless surveillance system, *J. Supercomput.*, vol. 73, no. 3, pp. 973–992, 2017.

[3] A. Koutsia, T. Semertzidis, K. Dimitropoulos, N. Grammalidis, and K. Georgouleas. Intelligent traffic monitoring and surveillance with multiple cameras, in *Proc. 2008 Int. Workshop Content-Based Multimedia Indexing*, London, UK, 2008, pp. 125–132.

[4] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. K. Zhang, et al., End to end learning for self-driving cars, arXiv preprint arXiv: 1604.07316, 2016.

[5] S. Haji and A. Varol, Real time face recognition system (RTFRS), in *Proc. 2016 4th Int. Symp. Digital Forensic and Security*, Little Rock, AR, USA, 2016, pp. 107–111.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Proc. 25th Int. Conf. Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[8] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, RFAL: Adversarial learning for RF transmitter identification and classification. *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 2, pp. 783–801, 2020.

[9] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, Physics-informed neural networks for high-speed flows, *Comput. Methods Appl. Mech. Eng.*, vol. 360, p. 112789, 2020.

[10] Q. Wang, J. R. Hopgood, N. Finlayson, G. O. S. Williams, S. Fernandes, E. Williams, A. Akram, K. Dhaliwal, and M. Vallejo, Deep learning in ex-vivo lung cancer discrimination using fluorescence lifetime endomicroscopic images, in *Proc. 2020 42nd Annu. Int. Conf. IEEE Engineering in Medicine & Biology Society*, Montreal, Canada, 2020, pp. 1891–1894.

[11] X. Li, M. He, H. Li, and H. Shen, A combined loss-based multiscale fully convolutional network for high-resolution remote sensing image change detection, *IEEE Geosci. Remote Sens. Lett.*, vol. 19, p. 8017505, 2021.

[12] B. Chesney and D. Citron, Deep fakes: A looming challenge for privacy, democracy, and national security, *Calif. Law Rev.*, vol. 107, pp. 1753–1820, 2019.

[13] H. Farid, Creating, weaponizing, and detecting deep fakes, https://www.usenix.org/conference/usenixsecurity19/presentation/farid, 2019.

[14] J. Lukas, J. Fridrich, and M. Goljan, Digital camera identification from sensor pattern noise, *IEEE Trans. Inf. Forensics Secur.*, vol. 1, no. 2, pp. 205–214, 2006.

[15] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, Source camera identification based on CFA interpolation, in *Proc. IEEE Int. Conf. Image Processing 2005*, Genova, Italy, 2005, p. III-69.

[16] K. R. Akshatha, A. K. Karunakar, H. Anitha, U.

Raghavendra, and D. Shetty, Digital camera identification using PRNU: A feature based approach, *Digital Invest.*, vol. 19, pp. 69–77, 2016.
[17] J. Bernacki, Robustness of digital camera identification with convolutional neural networks, *Multimed. Tools Appl.*, vol. 80, no. 19, pp. 29657–29673, 2021.
[18] D. Freire-Obregón, F. Narducci, S. Barra, and M. Castrillón-Santana, Deep learning for source camera identification on mobile devices, *Pattern Recogn. Lett.*, vol. 126, pp. 86–91, 2019.
[19] M. De Marsico, M. Nappi, D. Riccio, and H. Wechsler, Mobile iris challenge evaluation (MICHE)-I, biometric iris dataset and protocols, *Pattern Recogn. Lett.*, vol. 57, pp. 17–23, 2015.
[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in *Proc. 2015 IEEE Conf. Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 1–9.
[21] N. Sharma, V. Jain, and A. Mishra, An analysis of convolutional neural networks for image classification. *Procedia Comput. Sci.*, vol. 132, pp. 377–384, 2018.
[22] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proc. 13^{th} Int. Conf. Artificial Intelligence and Statistics*, Sardinia, Italy, 2010, pp. 249–256.
[23] G. Bradski and A. Kaehler, *Learning OpenCV*: *Computer Vision with the OpenCV Library*, Sebastopol, CA, USA: O'Reilly Media, 2008.
[24] K. P. Murphy, *Machine Learning*: *A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[25] L. van der Maaten and G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.
[26] R. Livni, S. Shalev-Shwartz, and O. Shamir, On the computational efficiency of training neural networks, in *Proc. 27^{th} Int. Conf. Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 855–863.
[27] K. He and J. Sun, Convolutional neural networks at constrained time cost, in *Proc. 2015 IEEE Conf. Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 5353–5360.

Chaity Banerjee earned the MS and PhD degrees in computer science from Florida State University both in 2017. Currently she is a postdoctoral associate in the Department of Industrial Engineering at the University of Central Florida. Her research interests include machine learning, data analysis, feature segmentation and localization in big data including Cryo-EM tomography and Single particle electron microscopy.

Tharun Kumar Doppalapudi earned the MS degree in computer science from The University of Alabama in Huntsville in 2021. His research interests include deep learning and its applications to real world problems.

Eduardo Pasiliao Jr. is a senior research engineer at the Air Force Research Laboratory Munitions Directorate (Eglin AFB FL) and the director of the AFRL Mathematical Modeling and Optimization Institute. He received the BS degree in mechanical engineering from Columbia University, ME degree in coastal and oceanographic engineering, and PhD degree in industrial and systems engineering from the University of Florida in 1992, 1995, and 2003, respectively. His research interest is in mathematical optimization with emphasis on social and communication networks.

Tathagata Mukherjee earned the MS and PhD degrees in computer science from Florida State University in 2014 and 2016, respectively. Currently he is an assistant professor in Computer Science at the University of Alabama in Huntsville. He has worked extensively with software defined radios with applications to assured communication and passive sensing. His research interests are broadly in the areas of cyber-security and network forensics where he is interested in applying machine learning, data analytics and optimization techniques for offensive security, malware analysis and Internet forensics. He is also interested in quantum computing and its applications, the theory and application of deep learning, and graph theory.