Camera Model Identification with Convolutional Neural Networks and Image Noise Pattern

Zhen Zuo[1]

***Abstract*— Camera source detection has drawn a lot of attention in past decade. It enables us to solve a wide range of problems, from crime evidence identification to photo tampering detection. In this paper, some main methods people used in this area for past decade will be reviewed in the Introduction and Method sections. Also in the Method and Result sections, I compared and improved state-of-the-art approaches to solve camera model identification problem. In the latter part, I proposed a novel approaches based on Convolutional Neural Networks and Image Noise Pattern. Results on the dataset from Kaggle shows that to identify source camera, Convolutional Neural Networks can be applied to the pattern noise rather than directly to the original pictures to achieve at least similar or even better performance.**

I. INTRODUCTION

Thanks to smart phone, around more than one trillion digital photos has been taken in 2017 [1]. Cameras have became the most important part of the phone recent years. On the other hand, graphics editors on smart phone also become popular and they are easier to use for common people, which brings a lot of troubles to digital investigations. Furthermore, technologies such as DeepFake [2] has successfully identify and swap faces in a picture or video and they look very realistic. A tampered or modified photo can be used for entertainment but should never be admissible in court. The question is how to prove a photo was taken by the certain device without modification? Camera model identification can help to answer this kind of problems. It can be used to identify the type of the device that taken an image, distinguish between devices of the same brand, or distinguish even among identical models.

Before performing camera model identification, it is necessary to be familiar with Image Acquisition Process. Figure 1 shows an simplified version of Image Acquisition Process. In Figure 1, the light from real word scene will be focused by lenses first and go through the optical filter to color filter array [3]. The lenses can change the direction of light and focus the light. Useless lights such as infrared ray will be removed from that filter. For the color filter array (CFA), the most common CFA is RGBW sensor. It is a matrix of pixels. Each pixel in CFA can only record the strength of one color out of green, red, and blue. How can one sensor generates three channel image? This is because Demosaicing Algorithm [4] is applied, which is an algorithm impute the missing values and make the raw one dimension matrix into a three dimensional RGB matrix. After reaching color filter array , the charge-coupled device (CCD) image

[1]Email: `zhenzuo2@illinois.edu`

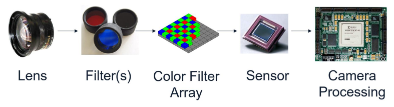


Fig. 1. Image Acquisition Process.

sensor will capture the color and in-camera software will process the color signal to form a raw digital image. After a series out-camera nonlinear processing, the final digital image will show up on the phone screen and save to the phone memory. During this process, the footprints will be left to the output image mainly caused by the CCD image sensor. Of course other process may also cause noise in the output image but the error caused by CCD are more unique and stable. CCD image sensor plays an important role in Image Acquisition Process because it translate incoming photons into electron charges which is a electronic signal. Normally, a phone camera has a 2D array of several million CCDs. For example, Huawei P20 has 40 million CCDs and each CCD is responsible for one pixel in the final image. So it contains a large amount of information. Theoretically, if each CCDs receives totally the same light as input, the output electron charges should be identical. However, nothing is perfect. The output electron charges are slightly different due to the small variations in the CCD size or substrate material. More specifically, the oxide of silicon on sensor is not totally flat due to the manufacturing error and it will cause some parts of the sensor are more sensitive than the other parts. This slightly difference will eventually cause the sensor noise in final image. Even the sensors for different devices from the same brand and even the same model can be different because of the manufacture error, which makes it possible to perform camera model identification to capture this kind of error.

II. METHOD

A. Image Feature

Using image features to do source camera identification was first proposed by Kharrazi etc. in 2004 [5]. The idea is to extract the features from image and use those features in a classification model to classify images. The process is done by the following steps:

Totally, there are 34 features extracted from the image.

1) Average pixel value: Those features are based on the gray world assumption. If a photo with enough color variations, the color of the average of the three RGB channel should be gray. So those features are the mean value of three RGB color channels.

2) RGB pairs correlation: Those features are three numbers measure the correlation between three color channels. Kharrazi believed different cameras have different color patterns so the correlation of color channel should vary. There are three features called RB, RG, and BG.

3) Neighbor distribution center of mass: The detail of this measure was described in the Kharrazi's paper. The basic idea is to find the neighbors for each pixel value for each color band first and obtain the distribution. From the obtained distribution, it can give us an indicator that how sensitive the sensor to different color intensity. The center of mass of the distributions are extracted as three features.

4) RGB pairs energy ratio: According to the original paper, those features were used to measure the white point correction. The calculated measures are: $E_1 = \frac{|G|^2}{|B|^2}$, $E_2 = \frac{|G|^2}{|R|^2}$, and $E_3 = \frac{|B|^2}{|R|^2}$.

5) Wavelet domain statistics: Three separable quadratic mirror filters are applied to each of three color channels. So finally nine features are selected.

After extracting all features, a classifier is applied to this Multi-class classification problem. Popular methods are Support Vector Machine, Quadratic Discriminant Analysis, and Random Forest classier.

Based on Kharrazi's idea, researchers tried to extracted additional features to improve this method. In 2009, Wang ect. extracted higher-order wavelet features [6] and wavelet coefficient co-occurrence features with feature selection method and muti-class SVM. Jeyalakshmi etc. in 2018 used Gray Level Co-Occurrence Matrix [7] as features. The features and classier may be different but the ideas among those paper are the same.

This method have some obvious disadvantages. First, the accuracy of this method drops quickly when the number of cameras increases. The accuracy of a binary classification problem is about 97% but when there are five devices, the accuracy is only about 80%. And when the number of cameras is 9, the confusion matrix is shown in Figure 2 and the accuracy is only around 60%. The reason is that even 34 features are exacted, there is still a large amount of information is lost after doing the features extraction. Also, it is pretty complex to extract all the 34 features. It is very time consuming to get all features from photos.

	S1	S2	M1	M2	N1	N2	N3	N4	L1
S1	**92**	4	0	0	0	0	0	3	0
S2	5	**63**	1	0	4	0	0	32	3
M1	0	3	**60**	11	3	1	3	3	5
M2	0	0	22	**67**	4	9	5	0	5
N1	0	6	2	3	**57**	3	6	7	18
N2	0	1	2	6	3	**68**	22	0	0
N3	0	1	7	10	1	18	**62**	1	1
N4	3	16	2	0	7	0	0	**36**	12
L1	0	6	4	3	21	1	2	18	**56**

Fig. 2. Image Feature Confusion Matrix.

B. Photo Response Non-Uniformity noise

This is the most popular method in past ten years. Photo noise method was proposed by Lukas in 2006 [8] and get more than eight hundred citations. In Figure 3, it shows the decomposition of noise pattern in an image. Pattern noise can be caused by Fixed Pattern Noise (FPN) and photo-response non-uniformity noise (PRNU). FPN primarily refers to pixel level differences when it is totally dark. FPN can be removed automatically for most of cameras nowadays. As described in former section, PRNU is the result of manufacturing imperfections and it is caused by the lack of homogeneity of imaging sensor's silicon area. The following attributes makes it a good candidate for source camera identification: (1) It is similar to white Gaussian Noise so it can contains noise information. (2) It is unique to every sensor even the same brand and the same model. This makes it possible to distinguish same model cameras. (3) Unlike other noise, PRNU is not affected by ambient temperature or humidity. It is much more stable than any other pattern noise.

The PRNU can be estimated by the method described as following.

$$I = I^O + I^O K + \Theta$$

where I^O is the original input image, I is the output image we got, K is the effect of the sensor pattern noise, and Θ is the random error term. To estimate K, first assume $I^O = F(I)$, where F() is the filtering process. Then K can be estimated with maximum likelihood estimate. Let

$$W = I - F(I) = I - I^O = I^O K + \Theta$$

Then the estimated K is

$$\hat{K} = \frac{\sum_{i=1}^{N} w_k I_k}{\sum_{k=1}^{N} (I_k)^2}$$

With $\hat{K}$, a correlation statistics is used to measure the similarity between the W and IK, such as peak-to-correlation-energy (PCE) value. With a certain $\hat{K}$, if $\hat{W}$ and $I\hat{K}$ are highly correlated, then this images may within the same group of images which $\hat{K}$ calculated from.

More paper published recent years with the similar ideas. In Tuama 2006 [9], noise pattern information was extracted by using high order statistics from POL-PRNU, which is contaminated PRNU. In Amerini etc. 2009 [10], it has been proved that different filters can play an important role in PRNU estimation. Many paper published to improve this method by using different filters, such as Content Adaptive Guided Image Filter [11] by Zeng 2016. There are also some paper improved the maximum likelihood estimation. Lawgaly ect. in 2014 found Weighted Maximum Likelihood Estimator has a better performance than simple maximum likelihood because luminance may also influence pattern noise.

This method is usually more accurate than the Image Feature Method but it also has some shortcomings. First, $\hat{K}$

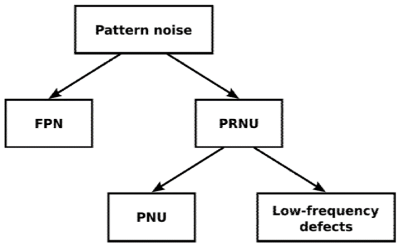


Fig. 3. Pattern noise of imaging sensors.

is very computational expensive. In the formula described above, the denominator of $\hat{K}$ is summation of N matrices. The dimension of each matrix is the same as the original image. Suppose the resolution of image is 20,000,000 pixels, which is common for most of phone cameras, the multiplication of two matrices with 20,000,000 elements are very expensive to compute (a few hours on my personal laptop). When there are N images, the multiplication will be repeated 2N times. So the number of images for each class is usually from thirty to fifty. It makes this method not working for large amount of photos, which is normal in real life.

C. Convolutional Neural Networks

Strictly speaking, Convolutional Neural Networks is one of the Image Feature methods. Two paper have been published in 2017 in this area. One is Bondi September 2017 [13] and one is Obregon October 2017 [14].

Bondi built a CNN model with four convolution layers. The final layer was followed by a ReLU layer to produce a 128 dimensional feature vector. SVM is applied to this feature vector of 128 elements. The dataset used in this paper is the Dresden image database [15]. It is a well-known dataset in source camera identification areas. It contains 18 different camera models and 83 scenes. The total number of images is more than 13,000. Each image was divided into 64×64 small patches for CNN model.

Obregon was more focused on explaining how Convolutional Neural Networks (CNN) works and some details in CNN architecture, such as Convolutional layer, Polling layer, and Fully-connected layer. For activation function, the performance of Leaky Rectified Linear Unit is compared with Rectified Linear Unit. Dropout rate and number of layers are tuned with cross validation. The dataset used in this paper is MICHE-I dataset. There are 3732 images from three different devices with different brands. Images were divided into 256 smaller patches with size 32×32 for the CNN model. The accuracy is about 98% for three classes.

D. Convolutional Neural Networks after PRNU

This is the method I proposed based on the methods above. In Pattern Noise method, one challenge is to estimate $\hat{K}$ from large scale matrix. The method CNN after PRNU is first using filter to get the original image and noise I just like patter noise patter. Then fitting the CNN on the noise pattern to perform the classification. This method should be more effective than the CNN method alone because the color information from the original images can be safely removed and only keep the Pattern Noise to identify the cameras.

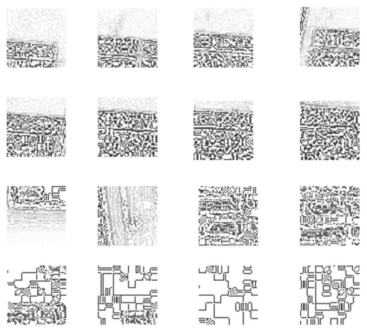


Fig. 4. Patches after Laplacian Gaussian smoothing filter.

This method may help people understand how Convolutional Neural Networks works in a more clear way. In the traditional PRNU method, when calculating estimated $\hat{K}$, only one channel of an images can be calculated each time. However, CNN after PRNU can work with colorful images without analyzing each color channel one by one.

III. DATASET

The dataset is from Kaggle. It is from a competition named ”Camera Model Identification”. Data can be downloaded with the link `https://www.kaggle.com/c/sp-society-camera-model-identification/data`. Then the data was simpled due to the limitation of computing power. Three out of ten cameras model are selected. They are iPhone-4s, Motorola-Droid-Maxx, and Samsung-Galaxy-S4. Because there is no way to get the labels of testing data, 275 training images were randomly divided into 200 images with training and all the other 75 images as testing.

The data was pre-processed the same ways as Bondi's paper. Each images was cropped into 64×64 patches without overlapping. After processing, there are 1,296,598 images in the training set and 496,657.

Another data was generated with the similar way. First, images were converted to gray images and then Laplacian Gaussian smoothing filter was applied to get the Pattern Noise. A sample can be found in Figure 4. This is filter for gray images.

Non-local Means De-noising was applied to the other data set. A sample can be found in Figure 5. This is the filter for colorful images.

There are three reasons to run the CNN with small 64×64 patches rather than the whole image: (1) It is more effective. The memory in GPU is usually than CPU and it cannot load the whole images unless with a very small batch size. More images can be loaded at the same time if images are only 64×64. Bondi has proved small patches with 64×64 can achieve nearly the same performance as the whole images. (3) It can generate more images for training.

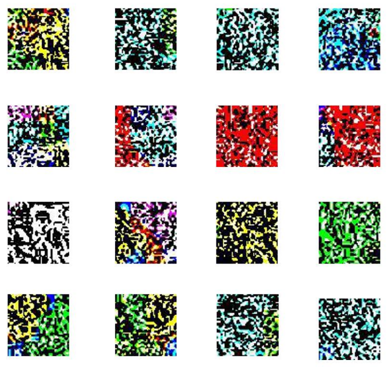


Fig. 5. Non-local Means De-noising.

IV. RESULT

The following results were from applying CNN to the Kaggle data. The Architectures of CNN is showed in Figure 6.

TABLE I
CONFUSION MATRIX OF CNN ON ORIGINAL IMAGES PATCHES

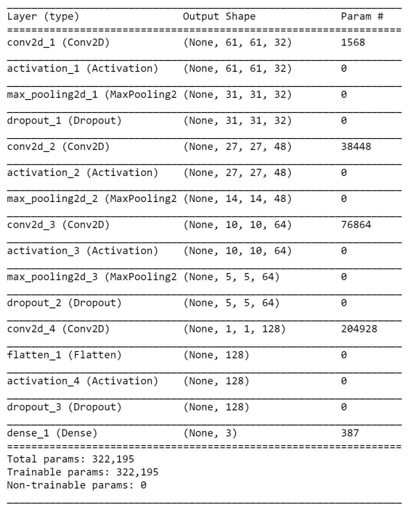
	iPhone	Motorola	Samsung
iPhone	34%	2%	0%
Motorola	0%	32%	0%
Samsung	1%	2%	25%

TABLE 1 is the confusion matrix of the CNN model. The accuracy is only about 91%, which is a little lower than the accuracy in the paper 94%. There are some possible reasons: (1) The dataset is different. The dataset I used was from Kaggle but the dataset Bondi used was Dresden Image Dataset. (2) The number of epochs are different. 100 epochs were run on my personal laptop with GeForce GTX 1050. 100 epochs took about 10 hours. In Obregon's paper, about 10,000 epochs were run. It is not surprised the accuracy is lower. (3) In Bondi's paper, not all small patches were included. Overly dark or saturated regions were excluded in both training and testing.

Similar accuracy achieved for both patches after Laplacian Gaussian smoothing filter and Non-local Means De-noising. The accuracy for Laplacian Gaussian smoothing filter is about 85% and the accuracy for Non-local Means De-noising is about 87%. It does not indicate those CNN after PRNU method generate worse performance than the CNN method because the model is not fully tuned and the number of epochs are not enough due to the limitation of computing power.

V. CONCLUSION

After applying the CNN to Kaggle dataset, we proved that CNN really can work to solve source camera identification problem. Based on the Image Feature Method, PRNU method, and CNN method, we prosed a CNN after PRNU method. Although the performance of CNN after PRNU method seems not as good as simply CNN in this experiment, but it may not the true case. Only two filters were selected to be tested and only 100 epochs were applied. It is hard to say what the performance will look like if more epochs applied and more filters tried. More experiments are needed in the future to make the final conclusion.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 61, 61, 32)        1568
_________________________________________________________________
activation_1 (Activation)    (None, 61, 61, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 48)        38448
_________________________________________________________________
activation_2 (Activation)    (None, 27, 27, 48)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 48)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        76864
_________________________________________________________________
activation_3 (Activation)    (None, 10, 10, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 1, 128)         204928
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
activation_4 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 322,195
Trainable params: 322,195
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6. Convolutional Neural Networks Architectures.

VI. CODE

Code can be found in `https://github.com/zhenzuo2/STAT578`. The ipython notebook includes the codes to process the data, CNN, and CNN After PRNU. Also a knitted version of code, which are html files are also available. The wight of the CNN model has been saved to a h5 file.

Since there are more than one million images after the data processing, only the raw data was uploaded to the Box. Data can be accessed with `https://uofi.box.com/s/6m5yfyl94tqkaqwgkc2d2vanb9v1om9l`

APPENDIX

REFERENCES

[1] Perret, E. (2017, January 19). Here's How Many Digital Photos Will Be Taken in 2017. Retrieved from `https://mylio.com/true-stories/tech-today/how-many-digital-photos-will-be-taken-2017-repost`

[2] DeepfakesFaceswap Retrieved from `https://github.com/deepfakes/faceswap`

[3] Color filter array. (2018, April 10). Retrieved from `https://en.wikipedia.org/wiki/Color_filter_array`

[4] Li, X., Gunturk, B., & Zhang, L. (2008). Image demosaicing: A systematic survey. Visual Communications and Image Processing 2008. doi:10.1117/12.766768

[5] Kharrazi, M., Sencar, H., & Memon, N. (n.d.). Blind source camera identification. 2004 International Conference on Image Processing, 2004. ICIP 04. doi:10.1109/icip.2004.1418853

[6] Wang, B., Guo, Y., Kong, X., & Meng, F. (2009). Source Camera Identification Forensics Based on Wavelet Features. 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. doi:10.1109/iih-msp.2009.244

[7] Jeyalakshmi, A, & Ramya Chitra, D . , Source Camera Identification using Image Features, 2018.

[8] Lukas, J., Fridrich, J., Goljan, M.: Digital Camera Identification from Sensor Pattern Noise. IEEE Transactions on Information Security and Forensics 1(2), 205214,2006

[9] Tuama, A., Comby, F., & Chaumont, M. (2016). Source Camera Model Identification Using Features from Contaminated Sensor Noise. Digital-Forensics and Watermarking Lecture Notes in Computer Science, 83-93. $doi : 10.1007/978-3-319-31960-5_8$

[10] Amerini, I., Caldelli, R., Cappellini, V., Picchioni, F., & Piva, A. (2009). Analysis of denoising filters for photo response non uniformity noise extraction in source camera identification. 2009 16th International Conference on Digital Signal Processing. doi:10.1109/ icdsp.2009.5201240

[11] Zeng, H., & Kang, X. (2015). Fast Source Camera Identification Using Content Adaptive Guided Image Filter. Journal of Forensic Sciences, 61(2), 520-526. doi:10.1111/ 1556-4029.13017

[12] Lawgaly, A., Khelifi, F., & Bouridane, A. (2014). Weighted averaging-based sensor pattern noise estimation for source camera identification. 2014 IEEE International Conference on Image Processing (ICIP). doi:10.1109/ icip.2014.7026084

[13] Bondi, L., Baroffio, L., Guera, D., Bestagini, P., Delp, E. J., & Tubaro, S. (2017). First Steps Toward Camera Model Identification With Convolutional Neural Networks. IEEE Signal Processing Letters, 24(3), 259-263. doi:10.1109/ lsp.2016.2641006

[14] Freire-Obregn, D., Narducci, F., Barra, S., & Castrilln-Santana, M. (2018). Deep learning for source camera identification on mobile devices. Pattern Recognition Letters. doi:10.1016/j.patrec.2018.01.005

[15] T. Gloe and R. Bohme, The Dresden image database for benchmarking digital image forensics, Journal of Digital Forensic Practice, vol. 3, pp. 150159, 2010.