Faking Sensor Noise Information

A Project Report

Presented to Dr. Chris Pollett

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements of the Class

CS297

By

Justin Chang

December 2021

ABSTRACT

Camera model detection has been a recent development in the field of image forensics. Photos taken by digital cameras leave identifiable traces imprinted on the resulting image. This report explores techniques on extracting these noise patterns and models that are available to categorize these noise patterns to identify a camera model type. Digital camera models have unique image processing pipelines that generate noise in a deterministic manner that can be used to identify the camera source. These fingerprints that are left behind can be estimated from a series of images of a camera and then be used to score test images with a similarity measure. The first deliverable details finding the sensor noise for pictures taken by an iPhone X. The second deliverable is an in-depth investigation on denoising methods using wavelet transformation. The third deliverable is an implementation of an existing benchmark to find the performance of modern models on this problem. The fourth deliverable is an implementation of a GAN in preparation for using GANs to model this unique problem. The fifth deliverable is an attempt to tailor the GAN to suit the needs of camera model detection. The goal for this master's project is not only detection of camera models, but the ability to fool detection models by sensor noise spoofing.

***Keywords* – Machine learning, computer vision, image forensics, Generative Adversarial Network (GAN), wavelet transform,**

TABLE OF CONTENTS

I. INTRODUCTION

The rise in availability of cameras in this digital age leads to copious amounts of pictures being taken that can be used as forensic evidence. Pictures taken of crimes and other events from the cellphones of bystanders could be used as evidence in court. However, with the widespread availability of picture and video alterations, the authenticity of images must be investigated before they can be used in court. One method that is recently being developed and has been used to authenticate images in some states is sensor noise fingerprint identification.

A digital camera has an imaging pipeline that produces noise on the resulting image. This noise is unique to each camera model and can be used to identify an image as being sourced back to a specific model of a camera. This type of authentication is useful for verifying images being taken by certain phones owned by different witnesses, defenders, or accusers in court. The methods for denoising an image vary, but the most common method found in research papers has been using wavelet transformations. These are chosen over their Fourier transform counterpart as they can detect more local features, which is imperative to detecting noise which is very small-scale differences between neighboring pixels.

After the generation of a sensor noise fingerprint for a certain model, a model would be needed to classify new images. Most of the work related to solving source camera identification involves creating the best classification model and various augments to the dataset to make the model more robust. Various machine learning methods seem to be the most popular methods since Machine learning is the new hot

topic. Neural networks and older methods like PCA are used for classification of these images. This master's project will attempt to create and GAN that trains a discriminator and a generator. The discriminator will determine whether the images are fake, and the generator will try to trick the discriminator with a generated image. The goal of this project is to use the GAN to take images from a certain camera, remove the noise and imprint noise that detection model would think it was sourced from a different camera.

II. DELIVERABLE 1: FIRST LOOK AT SENSOR NOISE

The goal for this deliverable is to first discover what a sensor noise pattern would look like from pictures with a camera firsthand. The first step was to take a series of pictures using a cellphone camera (iPhone X). A camera was set up and several pictures were taken of a black piece of carboard with the camera in a fixed position along with fixed position light source. The reasoning behind this setup was to capture the noise generated by the camera's internal components and minimize other sources of noise due to environmental factors. The goal was to generate some sort of fingerprint based off the noise from these images. In order to generate noise in this fashion the Python Image library (PIL) was used to process the images. All the images were processed and stored in red, green, blue (RGB) values of the image to a 3d array. A new 3d array was created that had entries that represented the average value of each pixel for each pixel location. That value was subtracted from each picture array to get a new array that represented the sensor noise pattern for each corresponding picture. This was the quickest rudimentary test that we came up with to get a sensor pattern.

This test was conducted under the assumption that there would be a majority of black pixels due to the average pixel values of all the pictures being almost identical and would deviate very little from each other. Thus, producing a delta from the average that is very small due to values in pixel arrays correspond to dark colors. However, the resulting pattern had more white pixels than expected. We assume this is because when we converted signed integers to unsigned integers, the two-compliment representation causes the conversion to turn small negatives into large numbers. This turns some spots white. On average some spots will be small numbers and other spots

will be large numbers (represented by white and black pixels). Also noticed larger blocks of pixels near the edges of the picture and finer grains of noise near the center. This indicates that the camera is more sensitive towards the middle of the lens. It is also possible that whiter images have less noise (larger blocks in the noise map). Upon comparing this experiment with another proposed by Lukas et al. [1], similar and different results/methods were discovered. Lukas et al. used pictures of a light box and used digital cameras with infinite exposure settings. The white balance was also set to gray to avoid saturation from overwriting noise. This method was to avoid dark current that is present in dark pixels from over influencing the noise gradient since dark current is more heavily influenced by environmental factors. Lukas et al. wanted the fingerprint of the camera to derive from component manufacturing deviations that are consistent among any images taken by the camera. One similar conclusion that we both noticed was the vignette of the image. The sensor seemed to be less sensitive on the edges of the screen than the middle. The paper did a lot more tests than this experiment covered and with different transformation functions that just subtraction from the average. The next deliverable will include attempts to use different analyzing functions on the source images to produce sensor patterns in Deliverable 2.

Figure 1: Example of Sensor noise

III. Deliverable 2: Transformation Functions And Wavelet Decomposition

The goal for this deliverable was to thoroughly research and practice using signal analyzing functions. Signal decomposition is used for denoising images. For this project, it is assumed that the images already have inherent noise and one goal of the image processing pipeline it to try denoising the image to obtain the noise from the subtraction of the original image from the noisy one. After obtaining these "noiseprints", experiments were run to discover the best parameters and wavelet types were best for denoising an image.

In order to reach an understanding about how wavelet decomposition work, a refamiliarization of the Taylor Series was needed. The purpose of transforming a function in this manner is that polynomial functions are much easier to compute, derivate and integrate, making them much easier to analyze. Fourier Transforms was the next step towards understanding the ground theory behind wavelets. The Fourier Transform is heavily used in signal analysis as it transforms a signal in the time domain to the frequency domain. It is similar to Taylor Series but instead of polynomials, the signal is decomposed into a series of sinusoidals. While we lose information in the time domain, we can approximate time using Fast Fourier Transforms which utilized a sliding a window of time and frequency. However, wavelets are typically much more accurate and decomposing a signal due to many different properties like having greater accuracy and a wide range in both frequency and time domain. Wavelets typically have compact support and can represent local regions in a signal much better. Wavelets also support a variety of window sizes that support large time windows for lower frequencies and narrow time windows for high frequencies. The theory behind wavelet denoising of an

image is the filter out small coefficients that represent small influences on the shape of the signal. In image terms, it means it smooths out the small variations in pixels which ultimately represent noise. The scipy-image library to denoise images and compare peak signal to noise ratio (PSNR) values between pairs of images to score each denoising method. Since there is no original image to compare against, measurements were done to see how consistent each method is. A comparison between each noise extraction method on how similar they are between different images was done, since the assumption is that the noiseprint should remain consistent among all phones no matter what the picture is of. The experiments proved that a biorthogonal wavelet filter with 3 and 5 vanishing moments for reconstruction and decomposition respectively is the most consistently accurate. The fewer vanishing moment in the filter produces a smoother image. Three different methods to generate noise were tested: scaled, absolute value, and unsigned integer. It was found that the scaled method has the highest average PSNR pair and determined that produces the most similar results.

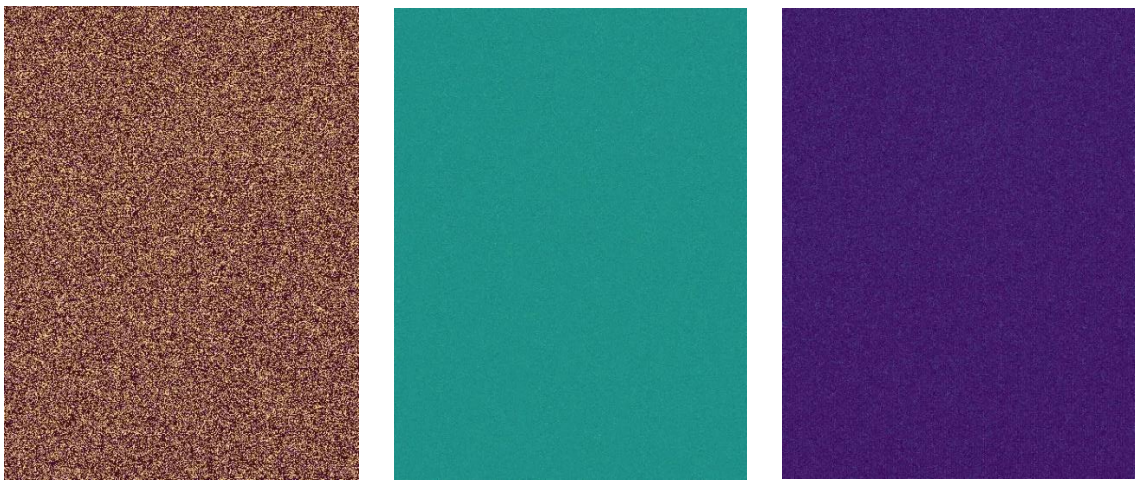

Figure 2:
Sensor noise prints from various extraction methods: left to right (abs, scaled, unit)

IV. DELIVERABLE 3: BENCHMARK SETUP AND TESTING

The goal for this deliverable is to explore another researcher's working code that creates noiseprints and uses them in camera model categorical comparison. The results of this code and will be used as a standard while working on custom models in CS298 and they will strive to surpass that standard. The original picture seems to be divided up. This can be interpreted as a method for data augmentation to get more data, but it can also be seen as attempting to identify local noise patterns in the images.

The python notebook was required to be modified in order to run on a local computer. This notebook was explained very well and intermediate outputs were printed to visually and statistically demonstrate intermediate results generated during the running of the model. The denoising parameter and methods as well as the noise reconstruction methods vary from the methods in deliverable 2. Later, a closer inspection of the parameters and methods will be done to improve models created for this master’s project. The machine ensemble section was not able to be fully functional and running correctly but it will probably not be used in this project as a benchmark.

The results were still not very good. Even with a CNN based approach, their accuracy only fell around 75% for their validation set. There were some limitations to the testing of their code. The original creator of the tutorial’s dataset was the only one used and a dataset with jpg compression like with phone images was not attempted to be used while running this notebook. There should be more concern about running a model on jpg compressed images due to the higher abundance of images taken with phones that use jpg compression. their code compared linear regression, k-means clustering, and NN approaches. They also used ensemble methods. Their NN approach performed

the best while the clustering approach performed the worst. A confusion matrix was generated from the sklearn library. After running this code, it was realized how important data visualization tools are towards analyzing results, especially during a presentation. Confusion matrices and graphs will be utilized to wherever they are applicable in the final report for CS298.

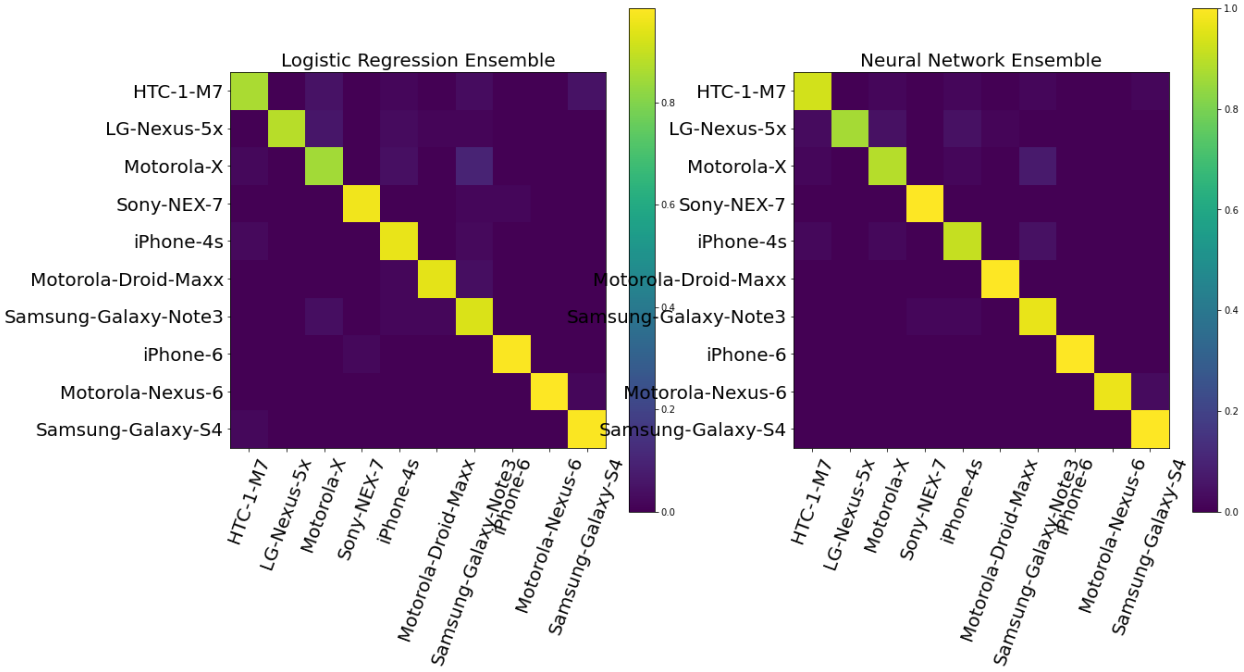

Figure 3: Example of confusion matrices from python notebook

V. DELIVERABLE 4: IMPLEMENTATION OF A GAN

While multiple methods for image denoising and classification of the noise residuals can be used, this project was taken with the assumption of generative adversarial networks (GANs) being the frontrunning model for tackling camera model identification and spoofing. Since the final goal of this project was to spoof camera sensor noise from one camera model to the other to fool a classification network, this seemed very similar to the GANs architecture. The GAN model architecture can be divided into two different submodels: the generator and the discriminator. These two models seem very useful for modeling the experiments that were considered of early in the brainstorming process of this project topic. The generator is the model that would create images with fake noise prints from non-original camera models. For this project, a conditional GAN (cGAN) is necessary because cGANs are able to allow for the specification of input parameters than make the output a little more deterministic. Specifically, the model would allow parameters to specify the type of camera model you would like to spoof. The discriminator submodel will classify the input into as real or fake. The classification part of GANs will be used to test out the quality of the noiseprint spoofs.

In CS298 the application of the concept of transfer learning to speed up the process of training the GAN would be highly desired, because training deep networks from scratch is very computationally expensive. Taking a popular image classification network like VGG. Several layers deep into VGG there are layers that are considered embedding layers. These layers will be the base for our generative network to insert the conditions that we specify to the generator. The rest of the layers will process and

upscale the vectors to output a image that is designed to feed into the discriminator and remain undetected as a fake. The discriminator network will also have to be designed in a way that corresponds to the input in the generator. Since the discriminator can only do binary classification of fake or real. We can only feed in images from the camera model that the generator is trying to spoof. Only after this is achieved, the discriminator will give a proper assessment of how the GAN is performing. The specifics of what type of networks how the hyperparameters of the discriminator and generator are still being decided.

An implement of a GAN based off a tutorial from tensorflow was conducted. The Modified National Institute of Standards and Technology (MNIST) database of handwritten digits were fed into the GAN. The tutorial displayed intermediate steps to demonstrate how the output of the generator improves over multiple cycles of training.

Figure 4: Example of images outputted by the generator after many cycles

VI. CONCLUSION

These deliverables improve my understanding of core principles that are needed to comprehend the project that will be created in CS298. Deliverable 1 explores the definition of a noise print and what it took to generate one. It was also a hands-on example on how to generate noise prints with locally captured images. Deliverable 2 explored the various method of denoising an image and creating a master noise print for a camera model. Wavelet denoising is the most popular method and seems to work very well. Deliverable 3 allowed me to survey the performance of current camera model detection networks and implement one to use to test another network against. Deliverable 4 gave me the opportunity to explore the inner workings of a GAN and the various type of GANs that may be suitable to achieve this project's goals.

For the final project the exploration PCA would be greatly beneficial due to very fast run times that don't scale exponentially due to dimensionality reduction. They also boast high accuracy, which invokes a little skepticism of due to the improvement in runtime and accuracy simultaneously. Using simple correlation measurements instead of a classification network plays a large role in their runtime speed up. An implementation of a cGAN on a local computer instead of using cloud resources is also desired since that is the model that is envisioned of performing the best.

REFERENCES

[1] J. Lukas, J. Fridrich and M. Goljan, "Digital camera identification from sensor pattern noise," in IEEE Transactions on Information Forensics and Security, vol. 1, no. 2, pp. 205-214, June 2006, doi: 10.1109/TIFS.2006.873602.

[2] D. Cozzolino and L. Verdoliva, "Noiseprint: A CNN-Based Camera Model Fingerprint," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 144-159, 2020, doi: 10.1109/TIFS.2019.2916364.

[3] S. Samaras, V. Mygdalis and I. Pitas, "Robustness in blind camera identification," 2016 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 3874-3879, doi: 10.1109/ICPR.2016.7900239.

[4] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp and S. Tubaro, "First Steps Toward Camera Model Identification With Convolutional Neural Networks," in IEEE Signal Processing Letters, vol. 24, no. 3, pp. 259-263, March 2017, doi: 10.1109/LSP.2016.2641006.

[5] R. Li, Y. Guan and C. Li, "PCA-based denoising of Sensor Pattern Noise for source camera identification," 2014 IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP), 2014, pp. 436-440, doi: 10.1109/ChinaSIP.2014.6889280.