Faking Sensor Noise Information

A Project Report

Presented to

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements of the

Degree of Science

By

Justin Chang

April 2022

The Designated Project Committee Approves

the Master's Project Titled

Faking Sensor Noise Information

by

Justin Chang

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2022

Dr. Christopher Pollett	Department of Computer Science
Dr. Robert Chun	Department of Computer Science
Dr. Mark Stamp	Department of Computer Science

ACKNOWLEDGEMENT

I would like to express my gratitude to Professor Chris Pollett for his guidance and encouragement during the creation of this master’s project. I would also like to thank the professor’s at SJSU for teaching me the necessary skills to complete this project. Lastly, I would like to thank my friends and family for supporting me in my journey for completing my Master’s of Science Degree in Computer Science.

ABSTRACT

Noise residue detection in digital images has recently been used as a method to classify images based on source camera model type. The meteoric rise in the popularity of using Neural Network models has also been used in conjunction with the concept of noise residuals to classify source camera models. However, many papers gloss over the details on the methods of obtaining noise residuals and instead rely on the self-learning aspect of deep neural networks to implicitly discover this themselves. For this project I propose a method of obtaining noise residuals ("noiseprints") and denoising an image, as well as a Generative model that can learn how to reproduce noise resembling a target digital camera model's noise noiseprint. Applying a noiseprint generated by this model onto a denoised image will be able to fool a discriminating model into classifying the wrong digital camera model. To the best of my knowledge, this is the first work that will explicitly detail denoising methods and noiseprint generation in a 128 by 128 resolution for specific camera models and individual cameras for the goal of fooling a classification model.

***Keywords* – Machine learning, computer vision, image forensics, Generative Adversarial Network (GAN), noise residual spoofing, denoising**

TABLE OF CONTENTS

I. Introduction

The rise in availability of cameras in this digital age has led 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. The goal of this project is to study and attack sensor noise fingerprint classification models.

Before noiseprints were introduced [1], there were attempts to reliably identify source camera models. EXIF headers data was a common distinguishing feature, however these data can be easily erased, as it is not tied to the image pixel data itself. Digital camera companies also made attempts to include digital signatures in the form of watermarks, biometric data of photographer, and hashes of images. These attempts did ensure integrity and/or authenticity of the image, however they required to be taken by special cameras. Most images won’t have these properties and thus will not be secure enough to use in the court of law. Several other methods that analyze picture features were also proposed, but they lacked the accuracy or failed to hold up under jpeg compression. Then the paper by Lucas et al. [1] that revolutionized source camera model identification was created. This paper was very influential to the point where most papers that about source camera model identification reference that paper. This project that I propose also makes use of camera sensor noise.

The digital camera imaging pipeline introduces noise that is specific to each camera model. Scene invariant noise is captured by models and used for classification. 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.

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. If spoofing of noise proves to be robust, this points out serious flaws in the sensor noise-based classification models.

This report is structured in the following order: The background will provide preliminary information that is required to understand the concepts that model attempts to incorporate. The preliminary work section acknowledges previous research papers that discuss topics related to sensor noise and source camera identification. The model design and implementation section discuss the details and reasoning behind the design choice for my model. The experiments and results section discuss the dataset and the results of my experiments. To conclude the report, future works are discussed along with concluding remarks.

II. Background

This section will discuss various background material related to sensor noise and camera model identification. An overview of the digital imaging pipeline will be given. Various denoising methods and their intended results will be discussed in the context of obtaining sensor noise and denoising an image. Popular methods for image classification will be discussed. Methods for preprocessing data for images to increase speed, accuracy and robustness will also be discussed. The concept of GANs will be introduced including, how to train them, and what they provide to my project.

A. *Digital Camera Pipeline*

Digital Cameras have an image processing pipeline that converts incoming rays of that that pass through the lens of the digital camera into digital bits of 0’s and 1’s for computers to interpret and display. However, this process cannot convert rays of light perfectly into bits and will introduce noise. This noise has been claimed to be deterministic and unique to a specific camera model. The pipeline first starts off with the scene in the real-world reflecting rays of light and some of those rays pass through the camera lens of the digital camera. Light rays that pass through the lens of the camera encounter mosaic of color filters that cover the imaging sensor. These sensors do not differentiate between wavelengths and are more sensitive to intensity. Therefore, each pixel sensor is only able to interpret the intensity of one specific color. The pattern and selection of colors in this mosaic are placed in a strategic fashion so that the digital processing pipeline can convert these single-color pixels into a full color image. The digital processing of the output of the color filter array includes, lens distortion correction, white balancing, brightness and gamma correction, jpeg compression, and demosaicing. At every step of this pipeline noise can be introduced. This noise has been determined to be manufacturing specific and therefore has been the focus of utilization for classification models.

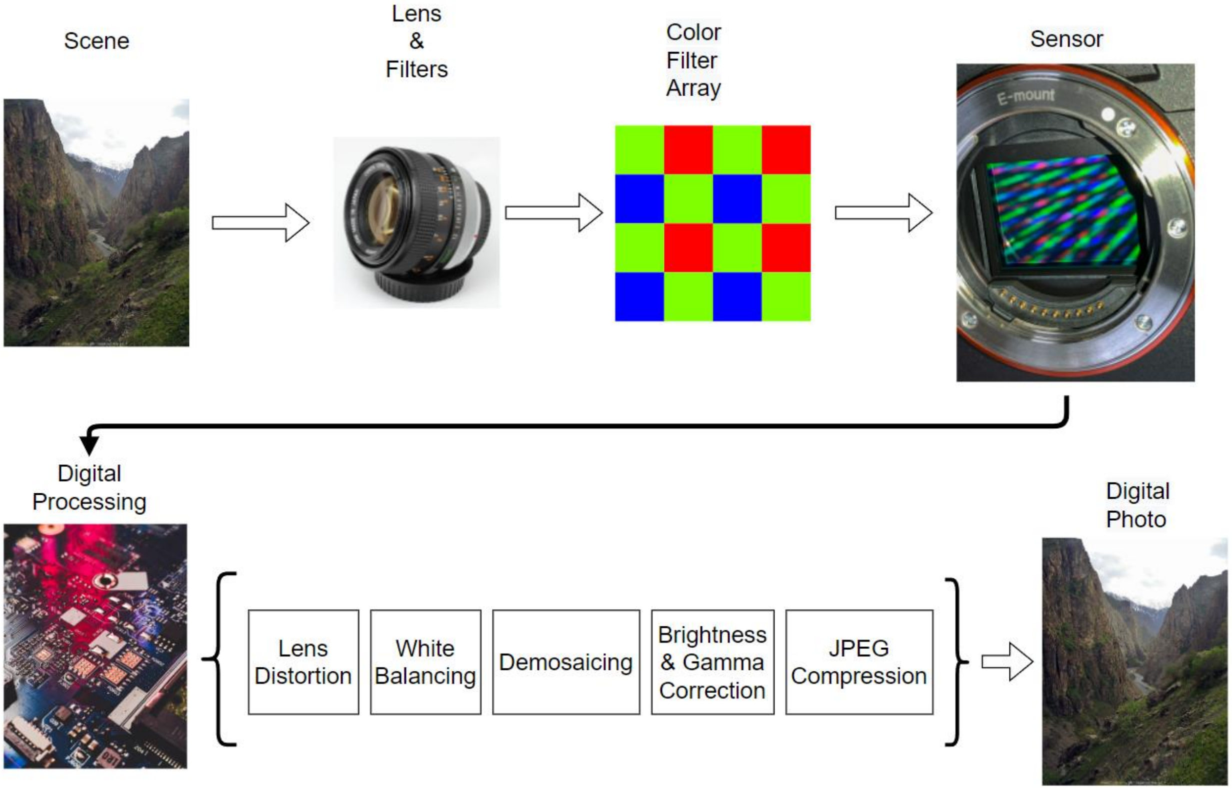

Figure 1. Digital Camera Pipeline. Sources from left to right and top to bottom [9][10][11][12]

B. *Noiseprints*

While there is noise in every image, it is important to note that for classification purposes, scene specific noise is not desirable to capture for classification models. Requiring each camera to take pictures of the same image at the same exact lighting, angle, and camera settings are very restrictive conditions for a model. Data would have to be specifically gathered from phones with this purpose in mind. Capturing only scene invariant noise allows the model to accept a much larger range of data as input data and will increase robustness. Pattern noise can be classified as Fixed Pattern Noise (FPN) and Photo-Response Non-Uniformity (PRNU). FPN are caused by dark currents. Dark current is the flow of electrons through the imaging sensor in the absence of light. The flow of electrons cause noise in the image and is also scene variant due to the scene requiring to located in a dark location. PRNU is generated from light sensitivity variations between each pixel. Slight variations occur between the silicon wafers that make up the sensors. PRNU noise is more consistent between all pictures and is not affected by environmental factors like FPN, such as temperature and humidity. Pixel non-uniformity is responsible for most of these qualities of PRNU and defective pixels make up the rest of PRNU. There are other sources of PRNU such as dust particles on the lens and zoom settings, however these are not consistent sources of noise and should not be considered by the model. PRNU noise patterns also tend to have a vignette shaped pattern, where the edges of the pictures are noisier than the center. While researching about the capture of noiseprints, tests where conducted on my own phone that confirmed and illustrated qualities of PRNU. A sample of a noiseprint is shown in Figure 3.

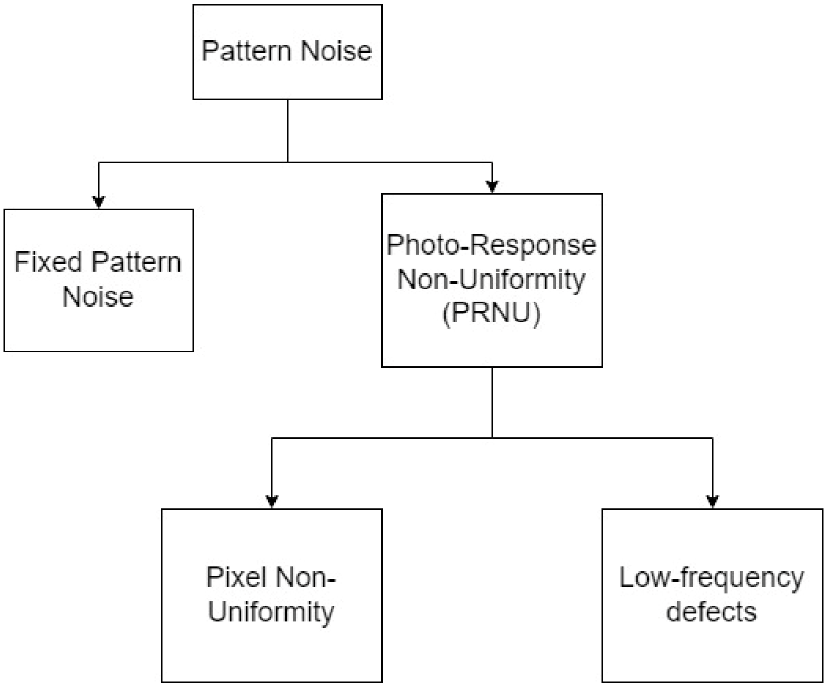

Figure 2. Pattern Noise Types

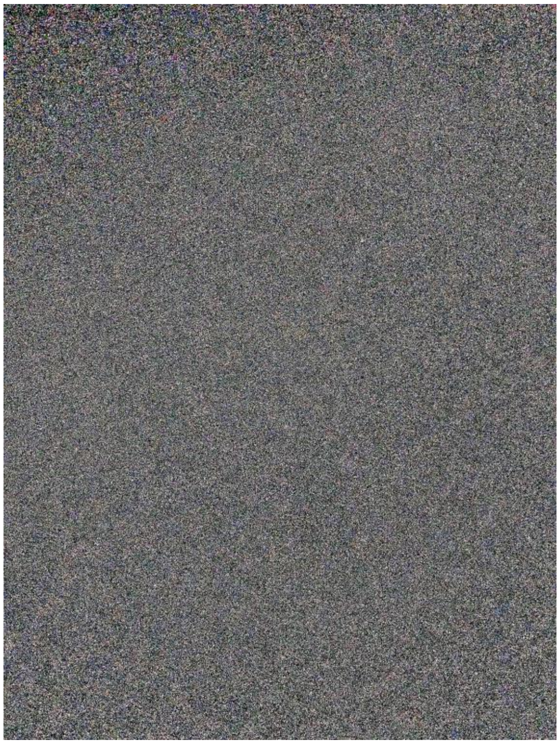

Figure 3. Noiseprint for iPhone X

C. *Denoising Filters*

Denoising is the reconstruction of an image that does not contain small neighboring pixel fluctuations that are attributed to noise. This concept can be traced back all the way to a simple approximation function called Taylor series. Fourier transforms are another more advanced version of approximation functions and finally people settled on wavelet transforms to be the golden standard. These techniques were first used for signal analysis with the purpose of representing discontinuous functions as a very similar function that is easier to differentiate. Being able to differentiate a function was crucial for the analysis of the function. Taylor series takes non polynomial functions and represents then with a similar looking polynomial function. Fourier transforms attempt to model waveforms with sinusoidals to decompose signals into tones of various frequencies. The disadvantage of both Taylor and Fourier is that they capture global frequency information and might not represent the original signal well.

Wavelet decomposition was an appropriate solution to capture local information since wavelets are not global functions and can capture local features of varying degrees. It can be seen as sliding a window across an image and capturing the essence of the image in each new section the window slides over. This technique can also be called wavelet denoising as the resulting wavelet transform does not contain small fluctuations that would be considered noise. The image that has been denoised will look smoother but still retain details of varying degree.

Median filters are also used in image smoothing because of their simplicity and their ability to preserve edges during the removal of noise. This filter assigns pixel values based on the median of the surrounding neighboring pixels. However, this type of filter has been known to leave scene related traces around edges that result in the misclassification of images. Many other filters are also considered by various researchers, however most papers often default to using Wavelet based filters as they produce the best results.

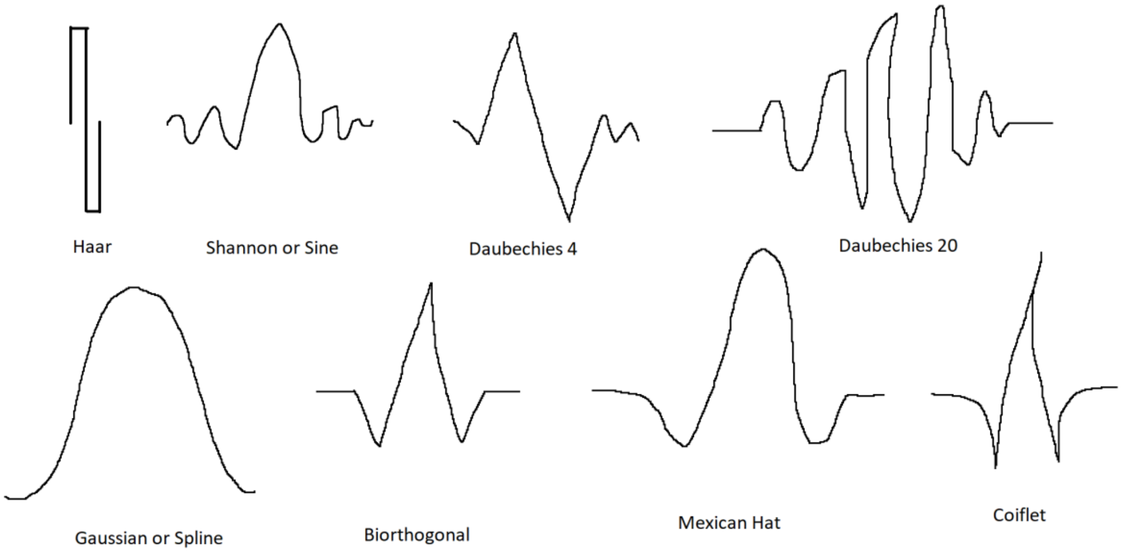

Figure 4. Examples of different types of Wavelets

D. *Local Binary Pattern (LBP)*

LBPs are an image texture descriptor that processes data to convert specific textures to more distinguished numerical features. This technique is also commonly combined histogram oriented gradients into order classify images based on histogram similarity. An example of a LBP method can be described as thresholding of neighboring pixels. If a pixel had a value of 100, all adjacent pixels would have a binary digit associated with that position. All adjacent pixels that are greater than the center pixel of 100 would get a their binary digit value flag to be 1. Adjacent pixels less than 100 would have a binary value of 0. The resulting binary number would require little space to store and contain a compressed representation of the nearby pixel texture. The histogram of LBP windows on an image can classify certain features within an image, for example classifying the nose in a picture of a face. The LBP window could also be the whole image if classifying the whole image is the desired outcome. This texture operator is analogous to a Convolutional Neural Network’s (CNN’s) sliding window filters. Sliding windows look for local textures to classify the image. However the power of the method of LBPs is that they are faster and can be used as a preprocessing step to augment data prior to being entered into a model.

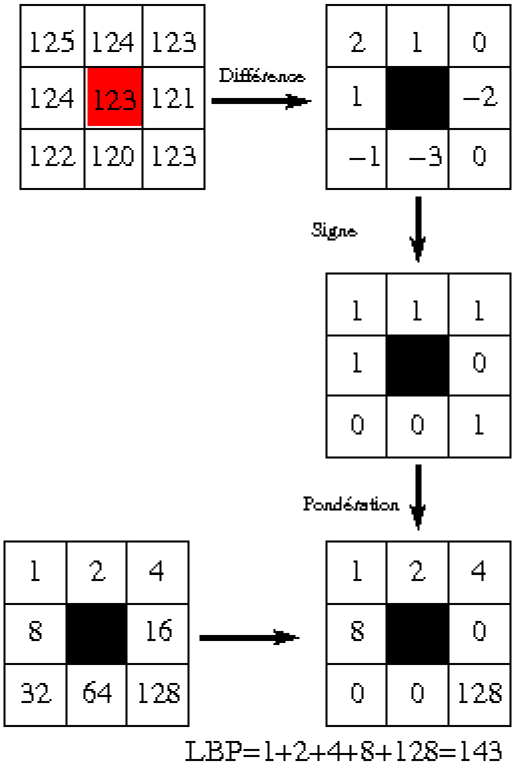

Figure 4. Examples of LBP calculation for a pixel [13]

E. *Convolutional Neural Networks (CNNs)*

A CNN is a network that uses the concept of sliding windows to analyze data. These windows are convolutional filters that analyze input features in batches as they move across the data. These filters can be seen as a form of regularization that prevent overfitting like fully connected multilayer perceptrons. This type of network is very effective and efficient at analyzing the local spatial patterns. Images typically require a more localized analysis for pattern recognition. CNNs are a very popular model for situations where the input data are images due to the network determining the optimal filters automatically, compared to other models needed engineers to custom make filters. In the case of RGB images, each color channel is treated with a different 3D convolution layer.

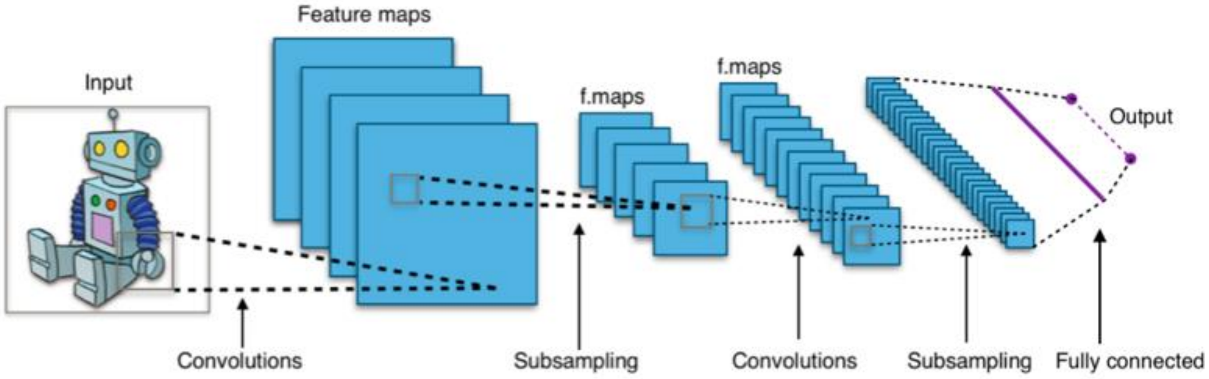

Figure 5. Example of a Convolutional Neural Network [14]

F. *Generative Adversarial Networks (GANs)*

The GAN model architecture can be divided into two different sub-models: the generator and the discriminator. Research has shown that generative models are an excellent method to train a model from smaller training sets, while discriminative models are more accurate in classification if the training set is large [3]. GANs are trained in tandem using the improvements of one model to train the other. The generative model's goal during training is the increase the rate at which the discriminative network incorrectly classifies the fake sample data that the generative network feeds into the discriminative network. The discriminative network's goal is the minimize the error rate at which the fake data is classified. Each network takes turns training once one of them reaches a certain level of success they switch.

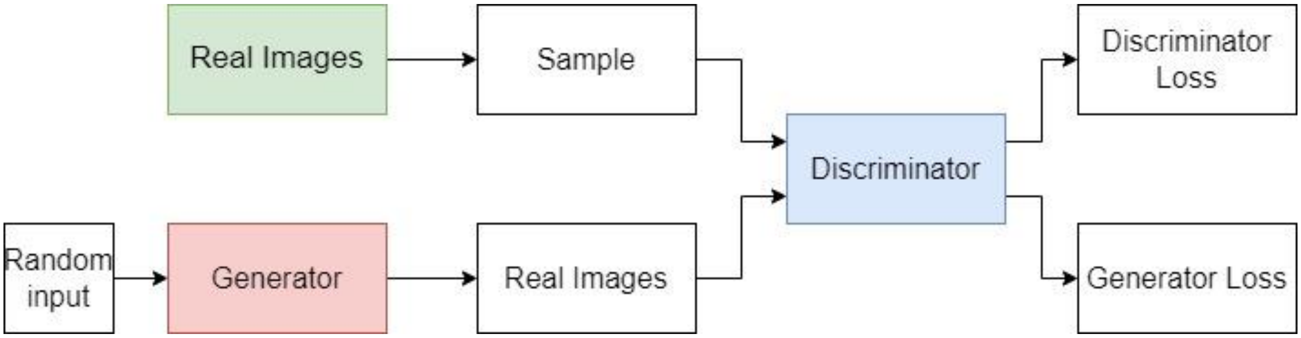

Figure 6. Visual Depiction of a GAN

III. RELATED WORK

A. *Sensor Pattern Noise discovery*

The paper “Digital Camera Identification from Sensor Pattern Noise” by Lucas et al. [1] was the first paper to approach the issue of digital camera identification through sensor noise and made a strong enough impact in this problem scope where almost every paper regarding camera identification references this paper. The authors of that paper only mentioned one other paper that discussed using sensor noise and they was a paper by Kurosawa et al [4]. In that paper they extracted dark current noise to classify camera models. However, this research required the datasets to contain images with large amounts of dark pixels. In addition, PRNU is a more robust component of sensor noise that is more pronounced and is more constant from a variety of scenes, leading to a more robust classification model. The discussion of noise print extraction largely goes unexplained in most papers regarding camera identification, while Lucas et al. [1] does go in depth. Instead of neural network models, which in 2005 probably wasn’t as popular to use as currently, detection by correlation was used. Different denoising methods were used to extract noise and they found that wavelet based denoising does the best job. They modified a wavelet denoising package called wavelab produced by Stanford that is not publicly accessible. The noise residuals extracted from each class of images are averaged to create a noiseprint for a camera model.

B. *Insertion of camera noiseprints onto artificially generated images*

Cozzolino et al. uses a GAN is used to generate images that are very similar to a photo that would be taken with a digital camera [5]. Then he inserts camera fingerprints from a specific camera model onto artificially generated images to fool a camera model identifier. This is the first instance of superimposing spoofed sensor noise onto artificially generated images and provides insight on the weakness of deepfake detection models.

C. *Arbitrary attacks on discriminative networks*

Chen et al. [6] discovers that modern camera model identification is weak to adversarial perturbations. Anti-forensic networks are created using GANs with the goal of confusing CNN-based classification models. Through a series of experiments, both white box and black box attacks on these classification models are proven to be successful at dramatically lowering it’s accuracy while still maintaining reasonable PSNR and Structural Similarity Index levels (SSIM). PSNR and SSIM are similarity scores that can be given to compare two images. While this paper does not conduct targets attacks designed to spoof a camera model, attacks that render a model to misclassify a model are still an important flaw to highlight.

D. *Tests in robustness of camera identification*

In the paper “Robustness in Blind Camera Identification”, Samaras et al. [7] analyses the effect that image alterations have on these models. Certain post processing methods images are applied and ran through the classification model. Some examples of these image manipulations are changes in white balance, gamma correction, contrast enhancement, and histogram equalization. The results of tests proved that models do hold up to these global image alterations, however the authors warn of the PRNU related manipulations, such as removing the PRNU factor, that are undetectable by human judgement drastically lowers accuracy. The model was also tested for training sample size requirements for good PRNU estimates. A recommendation of 100 sample images per class was advised for average wavelet-based denoising.

E. *PCA based denoising of noiseprints*

Li et al. [2] applies PCA-based denoising to the noiseprints themselves. This is expected to reduce the computation time of classification by reducing dimensionality. PCA-based denoising also has the effect of making the model more robust by suppressing irrelevant parts of the noiseprint and allowing the significant information to be valued more by classification models. They use a similarity measure based on simple correlation estimates, however these discoveries should hold up to neural network classification based approaches.

IV. Dataset

For this project, the goal was to simulate an attack on someone's personal device. The attacker would download the victim's images off of their phone and attempt to model the noiseprint for that particular phone. With this goal in mind, the total images available for training and testing had to be withing a reasonable number of images that would be on user's phone at any given time.

Three sets of phones were used to gather data. One Apple iPhone X and two different Samsung Galaxy S8s. Some of the data was gathered from images existing on the phone already and some extra pictures were taken to even out the sample sizes of all classes. A wide variety of images were taken in different conditions. Some images were taken outside in low and high levels of scene brightness. Scenes that were captures indoors were also taken. Pictures were taken in scenes of various temperatures and humidity as well since they are both known to affect noise. Pictures were taken with the base zoom level on all phones, but also were taken at varying physical distances from the subjects. Subjects of images differed from humans, animals, trees, buildings, etc. This also guaranteed that the distribution of colors within pixels were varied ensuring an accurate model of noise, since noise is also known to be affected by light source color. It is important to note that there were no image altering filters applied to these images.

In an attempt to follow the guideline set by Samaras et al., each model would have over 100 images per class to train on. The GAN and the classification network also did not share datasets to prevent possible cross contamination or overfitting. The

GAN had 240 images to train the generator and the discriminator on. The classifier used 275 images for its training set and 30 images were reserved for the test set.

Figure 7. Examples of images from each camera (left to right: iPhone X, Galaxy8c, Galaxy8l)

V. Model Design and Implementation

A. *Preprocessing for the GAN*

The GAN's generator is designed to model the noiseprint for each class, so the input images would need to have their noiseprints extracted to feed into the discriminator as training data. Each image had a center crop of 128 by 128 pixels taken from it. Previous works have also worked well with small crops taken from larger images. The center crop was put through a denoising filter. The original image was then subtracted from the denoised image to obtain the noiseprint for that particular image. The nose prints are arrays of 128 by 128 by 3 and are small negative and positive floats. These arrays are saved to be use for training the GAN. Multiple crops per image do result in higher accuracies for classification but each generator would have to train on crops respective to their position, thus greatly increasing training time. Typically, data is scaled before being used in models, however in order for the GAN to accurately reconstruct sensor noise, any scaling would produce a noiseprint that is magnitudes larger or smaller than the original image. This difference would cause the spoofing of the camera model to be unsuccessful as there would be a difference in the noiseprints between the training set and the spoofed testing set of the classifier.

B. *Training the GAN*

Each camera model would have its own GAN due to the need for three distinct generators. The 240 noiseprints for each class were stored as NumPy arrays in the npz format and can be fed into the GAN's discriminator as NumPy arrays. Random noise is the input for the discriminator portion of the model. The generator has three alternating layers of 2D convolutional layers, batch normalization, and leaky ReLU. The final output shape would be a 128 by 128 by 3 array that simulates attempts to emulate a noiseprint of that respective class. The discriminator has two alternating layers of 2D convolutional layers, leaky ReLU, and dropout. After those two alterting layers, the output is flattened and put through a 1-dimensional dense layer for classification of real or fake. The loss function for the discriminator rewards high accuracy of the real and fake images. The loss function for the generator penalizes correct classification of the generated fake samples. 50 epochs and a batch size of 20 are used in this model. When the model is finished training, their weights are saved so the generator can generate fake noiseprints for classification.

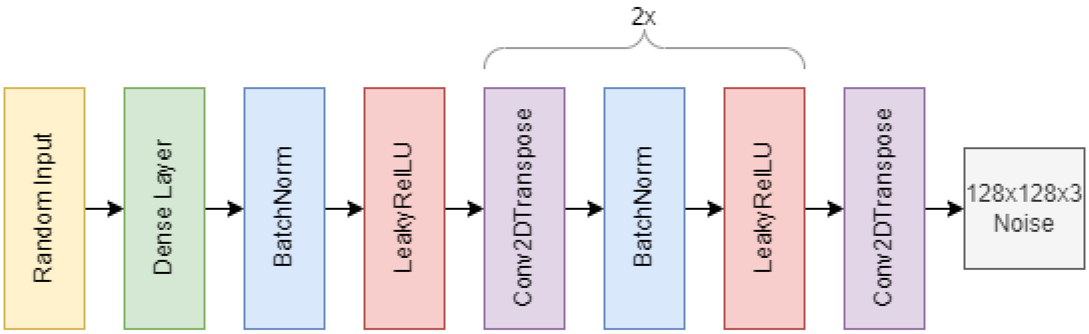

Figure 8. Diagram of the generator component of the GAN

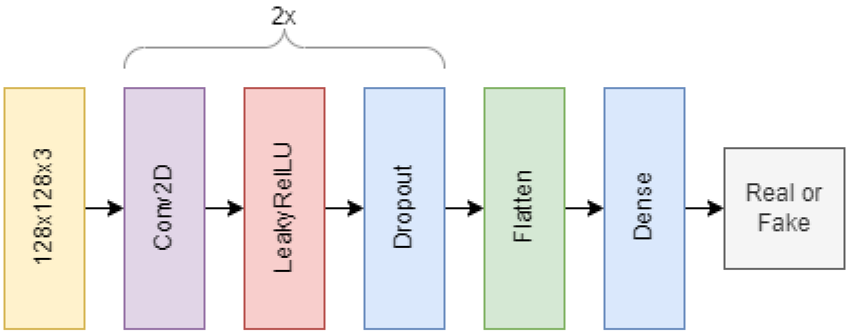

Figure 9. Diagram of the discriminator component of the GAN

C. *Preprocessing the Classifier*

Preprocessing the training data for the classifier involves taking a center crop of 128 by 128 for each image. The test data contains a mix of real and fake images. The real test images are also 128 by 128 crops. However, the fake images must be created by denoising the image, loading the GANs, and using the respective generator to apply a noiseprint to the denoised image. Various tests were done at this stage that will be covered in the next section. The 128 by 128 crops went through additional preprocessing that involved converting the 3D RGB pixel arrays into 3D LBP data. For histograms were generated for each color channel for 26 buckets creating a 78 element long array. Then the image is put through a wavelet denoising method (haar) and for each color channel. Three coefficients (arrays of 64x64) per moment per color channel are saved. Then the nth moment of the mean of those three coefficient arrays are calculated for 9 different moments and appended. This generates an additional 3 (RGB) * 3 (coefficients) * 9 (moments) = 81 element long array. The 78 elements and 81 element long array are concatenated and fed into the classifier to represent features for that image.

D. *Training and testing the Classifier*

Multiple models were used to classify the data. However, the MLP model and the 1D-CNN model produced the best results. The MLP has two layers the first layer being 256 nodes and the second layer being 128 nodes. The 1D-CNN consisted of 2 1D convolution layers with a kernel size of 3 by 3. They are followed by a dropout, max pooling, and dense layers.

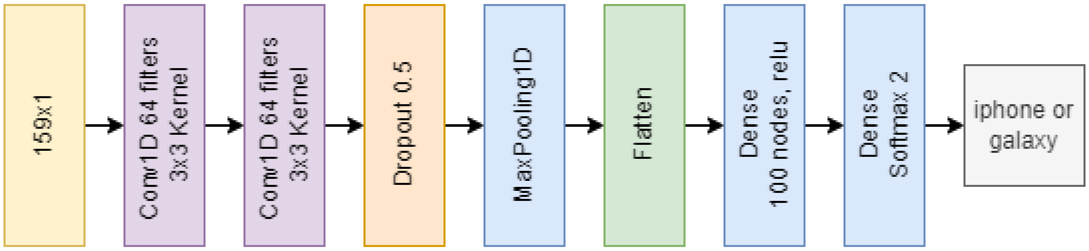

Figure 10. Diagram of 1D-CNN classifier

Logistic regression and K-means models were also tested and produced decent results. Hyperparameters for those models weren’t analyzed in great detail and therefore the structure of these models won’t be discussed as they were very simple.

A 2D-CNN model was also used as a classifier but, the preprocessing techniques for the input data of this model were different. They will be described in more detail in the experiments section of this report. there are 3 repetitions of 2D convolutional, ReLU, and max pooling layers. These are followed by the last flatten and dense layers.

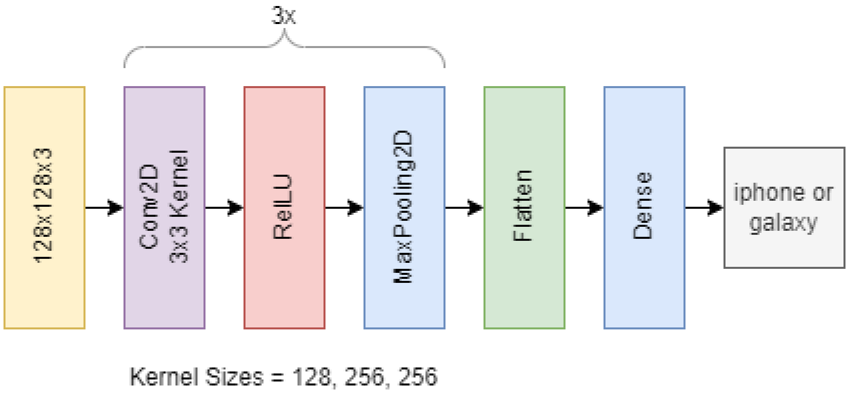

Figure 11. Diagram of 2D-CNN classifier

VI. Experiments and Results

Most experiments were conducted on 2 classes, iPhone X and Galaxy8c and unless specifically mentioned, the experiments below were conducted on this basis.

A. *Core set of experiments for performance metric*

The results were fed into a variety of core experiments that determined the quality of the changes. The experiments are listed below:

Experiment 1: non-spoofed images

Experiment 2: denoised images with no noise spoofing

Experiment 3: iPhone noise spoofing on all denoised images from both classes

Experiment 4: galaxy noise spoofing on all denoised images from both classes

Experiment 5: iPhone noise spoofed images only for denoised galaxy images

Experiment 6: galaxy noise spoofed images only for denoised iPhone images

Expected results are listed below:

Symbols in table are ideal results:

⋀ means a high number

⋁ means a low number

- means they are all equal

cropped: normal image that is unaltered

denoised (dn): denoising filter applied but no noiseprint applied

in-iphone: iPhone denoised image with iPhone spoofed noise added

in-galaxy: galaxy denoised image with iPhone spoofed noise added

gn-iphone: iPhone denoised image with galaxy spoofed noise added

gn-galaxy: galaxy denoised image with galaxy spoofed noise added

Actual

		cropped-iphone	cropped-galaxy8c
Predicted	iphone	^	v
	galaxy8c	v	^

Actual

		denoised-iphone	denoised-galaxy8c
Predicted	iphone	-	-
	galaxy8c	-	-

Actual

		in-iphone	in-galaxy8c
Predicted	iphone	^	^
	galaxy8c	v	v

Actual

		gn-iphone	gn-galaxy8c
Predicted	iphone	v	v
	galaxy8c	^	^

Actual

		cropped-iphone	in-galaxy8c
Predicted	iphone	^	^
	galaxy8c	v	v

Actual

		cropped-galaxy8c	gn-iphone
Predicted	iphone	v	v
	galaxy8c	^	^

Table 1. Experiment ideal outcomes

Experiment 1 is the baseline that that determines how well the model performs on unaltered images. Experiment 2 was designed to show that once in image is denoised the classifier would have equal difficulty in classifying each class. The rest of the experiments should have high values for their corresponding spoofed class or high values for their correct class is not spoofed.

B. *Double JPG compression*

When the original image is cropped and saved as a jpg, the image undergoes jpg compression twice. This had an adverse effect on the accuracy of the training, validation, and test set of the classifier. To prevent from further altering the image, the cropped image was saved as a PNG file. The lossless compression should prevent the altering of any noiseprints. For example, the accuracy of the training set of the neural network classifier rose from 87% to 93% and the validation accuracy rose from 78% to 89%. Due to the nature of using a small sample size and a simple model, it would have a harder time ignoring small image alterations.

C. *Denoising metrics*

Various denoising methods were applied to test images and the trained classifier was tasked to classify them. The ideal outcome would be an even split between all classes due to the noise print being wiped out. Therefore, the denoising method closest to 50% accuracy with the least biased would be the ideal denoising method. From the table below. The denoise_tv_chambolle() method from the skimage library was chosen. This code in this library used the paper by Chambolle et al. [8] as reference.

		Actual		
		cropped-iphone	cropped-galaxy8c	Accuracy
Predicted	iphone	26	7	0.816667
	galaxy8c	4	23	

		Actual		
		dn-bi-iphone	dn-bi-galaxy8c	
Predicted	iphone	23	6	0.783333
	galaxy8c	7	24	

		Actual		
		dn-bior35-iphone	dn-bior35-galaxy8c	
Predicted	iphone	25	6	0.816667
	galaxy8c	5	24	

		Actual		
		dn-bior44-iphone	dn-bior44-galaxy8c	
Predicted	iphone	23	5	0.8
	galaxy8c	7	25	

		Actual		
		dn-coif4-iphone	dn-coif4-galaxy8c	
Predicted	iphone	23	6	0.783333
	galaxy8c	7	24	

		Actual		
		dn-coif8-iphone	dn-coif8-galaxy8c	
Predicted	iphone	24	7	0.783333
	galaxy8c	6	23	

		Actual		
		dn-db4-iphone	dn-db4-galaxy8c	
Predicted	iphone	23	5	0.8
	galaxy8c	7	25	

		Actual		
		dn-db8-iphone	dn-db8-galaxy8c	
Predicted	iphone	23	3	0.833333
	galaxy8c	7	27	

		Actual		
		dn-gaus-iphone	dn-gaus-galaxy8c	
Predicted	iphone	17	24	0.383333
	galaxy8c	13	6	

		Actual		
		dn-median-iphone	dn-median-galaxy8c	
Predicted	iphone	25	25	0.5
	galaxy8c	5	5	

		Actual		
		dn-n1-iphone	dn-n1-galaxy8c	
Predicted	iphone	27	25	0.533333
	galaxy8c	3	5	

		Actual		
		dn-sym4-iphone	dn-sym4-galaxy8c	
Predicted	iphone	23	3	0.833333
	galaxy8c	7	27	

		Actual		
		dn-sym8-iphone	dn-sym8-galaxy8c	
Predicted	iphone	23	4	0.816667
	galaxy8c	7	26	

		Actual		
		dn-tv-iphone	dn-tv-galaxy8c	
Predicted	iphone	20	21	0.483333
	galaxy8c	10	9	

Table 2. Denoising method comparisons

D. *Rounding*

The data type for images are stored in integers. This makes operations inherently lossy when the altered image, which is a float, needs to be saved again. When floats are converted to integers, the rounding function performed better than the flooring floats to integers.

E. *Weighting the noiseprints*

To explain why the model wasn't behaving ideally, varying levels of noise strength were tested. There was a strong trend where the model better classified iPhone noise when a small multiplier was applied to the noise. Galaxy8c noise had the opposite result with the model classification being more ideal the stronger the noise is. These findings alluded to the fact that the model associates a less noisy image with iPhone and noisy images with galaxy. The number after the header represents the noise multiplier.

Actual

		in-iphone	in-galaxy8c	in-iphone0.01	0.05	0.1	0.3	0.5	0.7	1.05	1.1	1.25	1.5	2	2.2
Predicted	iphone	12	8	20	21	20	19	15	14	10	11	12	12	13	15
	galaxy8c	18	22	10	9	10	11	15	16	20	19	18	18	17	15

Actual

		gn-iphone	gn-galaxy8c	gn-galaxy8c0.01	0.05	0.1	0.3	0.5	0.7	1.05	1.1	1.25	1.5	2	2.2
Predicted	iphone	16	8	21	21	20	19	18	16	10	10	7	4	5	5
	galaxy8c	14	22	9	9	10	11	12	14	20	20	23	26	25	25

Actual

		cropped-iphone	in-galaxy8c	in-galaxy8c0.01	0.05	0.1	0.3	0.5	0.7	1.05	1.1	1.25	1.5	2	2.2
Predicted	iphone	26	8	21	20	20	16	13	10	9	7	5	5	8	6
	galaxy8c	4	22	9	10	10	14	17	20	21	23	25	25	22	24

Actual

		cropped-galaxy8c	gn-iphone	gn-iphone0.01	0.05	0.1	0.3	0.5	0.7	1.05	1.1	1.25	1.5	2	2.2
Predicted	iphone	7	16	20	20	20	20	18	17	15	15	13	11	12	10
	galaxy8c	23	14	10	10	10	10	12	13	15	15	17	19	18	20

Table 3. Noise weight gradient comparisons

F. *Random Noise Injection*

Random noise was also added to the images at different magnitudes to see how the model interprets random noise. The cropped image was not denoised before noise injection in this experiment. The range of noise is denoted by the tuples in the headers with the first number being inclusive and the last number being exclusive. The noise was uniformly distributed. As more noise was applied, the iPhone class was chosen substantially more than the galaxy8c. This result can be interpreted as the iPhone image sensor introduces noise in a more uniformly.

		Actual					
		cropped-iphone	**cropped-galaxy8c**	**iphone (-1,2)**	**galaxy8c (-1,2)**	**iphone (-2,3)**	**galaxy8c (-2,3)**
Predicted	iphone	26	7	20	4	18	5
	galaxy8c	4	23	10	26	12	25

		Actual							
		cropped-iphone	**cropped-galaxy8c**	**iphone (-65,66)**	**galaxy8c (-65,66)**	**iphone (-78,79)**	**galaxy8c (-78,79)**	**iphone (-88,89)**	**galaxy8c (-88,89)**
Predicted	iphone	26	7	24	27	25	28	26	29
	galaxy8c	4	23	6	3	5	2	4	1

Table 4. Random noise injection comparisons

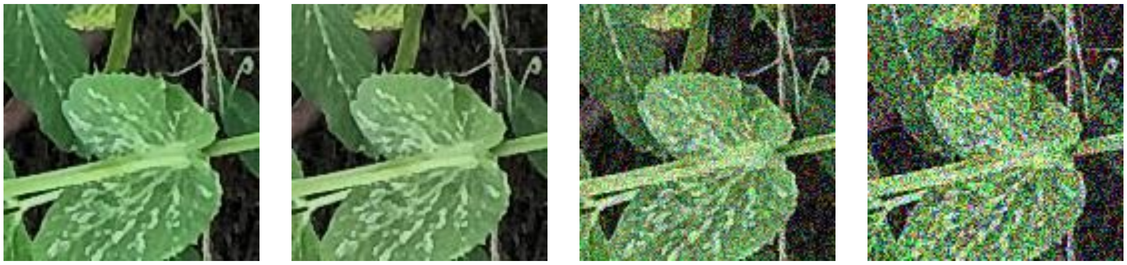

Figure 12. Images with different ranges of noise (original, pixel delta 16, 130, 200)

G. *Random Pixel Test*

Randomly generated pixels were generated to create a 128 by 128 by 3 image for this experiment. This test was created to analyze how the classification model would handle purely random pixels. All test images were classified as iPhone. The classification model seems to lean very heavily towards iPhone images when detecting a uniform distribution of pixels. This also confirms that the model thinks iPhone noise is uniform.

H. *Inter-Model Classification*

While the results were less than stellar for 2 class classification, it was still insightful to test three class classification. Two different phones of the same model (galaxy8) were used along with iPhone images. The results that were obtained were surprisingly good, as neither class was too dominant. The experiment 1 showed very promising results that suggest specific digital camera identification may not be too far into the future.

		Actual		
		cropped-iphone	cropped-galaxy8c	cropped-galaxy8l
Predicted	iphone	18	3	3
	galaxy8c	8	20	13
	galaxy8l	4	7	14

		Actual		
		denoised-iphone	denoised-galaxy8c	cropped-galaxy8l
Predicted	iphone	7	4	8
	galaxy8c	9	9	6
	galaxy8l	14	17	16

	Actual			
		in-iphone	in-galaxy8c	in-galaxy8l
Predicted	iphone	4	2	3
	galaxy8c	13	15	17
	galaxy8l	13	13	10

	Actual			
		8cn-iphone	8cn-galaxy8c	8cn-galaxy8l
Predicted	iphone	3	3	4
	galaxy8c	12	10	10
	galaxy8l	15	17	16

	Actual			
		8ln-iphone	8ln-galaxy8c	8ln-galaxy8l
Predicted	iphone	12	4	2
	galaxy8c	2	8	6
	galaxy8l	16	18	22

Table 5. Results of inter-model classification

I. *Random Noise Injection into Training Set*

Random noise was added onto the training set of the classifier to make the model more robust and improve results on the testing set. The results did not differ very much from the original experiments. The lower accuracy and misclassification of data in the original classifier trained on original images could have been attributed toward the overfitting of the model, due to the high training accuracy and the low test accuracy. This required augmenting the training data to add random noise. Two different magnitudes of randomness were applied, one with a delta range of 6 and one with a range of 2. Each range involved a uniform distribution with a mean of 0. This was an attempt to make the model more robust. The MLP model was chosen for this experiment. The results are shown in the figures below in this section. Noise injection that had delta variants of 6 point values for each color channel was typically higher than the standard noise print and thus gave us results that were not as accurate for non-altered test images. It also made the model more biased toward the galaxy8l class for the spoofed data. Noise injection with the 2 pixel delta experiment showed more accurate results for non-spoofed images and a bias towards iPhone images in spoofed data.

		Actual		
		cropped-iphone	cropped-galaxy8c	cropped-galaxy8l
Predicted	iphone	11	4	3
	galaxy8c	8	15	7
	galaxy8l	11	11	20

		Actual		
		denoised-iphone	denoised-galaxy8c	cropped-galaxy8l
Predicted	iphone	3	2	1
	galaxy8c	12	9	12
	galaxy8l	15	19	17

		Actual		
		in-iphone	in-galaxy8c	in-galaxy8l
Predicted	iphone	7	11	6
	galaxy8c	11	8	9
	galaxy8l	12	11	15

		Actual		
		8cn-iphone	8cn-galaxy8c	8cn-galaxy8l
Predicted	iphone	5	6	2
	galaxy8c	10	8	9
	galaxy8l	15	16	19

		Actual		
		8ln-iphone	8ln-galaxy8c	8ln-galaxy8l
Predicted	iphone	7	5	2
	galaxy8c	6	7	8
	galaxy8l	17	18	20

Table 6. Results of training on images with random noise injections (pixel range [-3,3]

		Actual		
		cropped-iphone	cropped-galaxy8c	cropped-galaxy8l
Predicted	iphone	18	7	6
	galaxy8c	6	15	7
	galaxy8l	6	8	17

		Actual		
		denoised-iphone	denoised-galaxy8c	cropped-galaxy8l
Predicted	iphone	13	16	10
	galaxy8c	6	4	5
	galaxy8l	11	10	15

		Actual		
		in-iphone	in-galaxy8c	in-galaxy8l
Predicted	iphone	15	13	10
	galaxy8c	7	7	6
	galaxy8l	8	10	14

		Actual		
		8cn-iphone	8cn-galaxy8c	8cn-galaxy8l
Predicted	iphone	14	16	14
	galaxy8c	4	6	6
	galaxy8l	12	8	10

		Actual		
		8ln-iphone	8ln-galaxy8c	8ln-galaxy8l
Predicted	iphone	18	10	3
	galaxy8c	2	5	5
	galaxy8l	10	15	22

Table 7. Results of training on images with random noise injections (pixel range [-1,1]

J. *PCA denoising on noiseprints*

Following the path of Li et al. [2], PCA based denoising methods were also applied to noiseprints to see if results improved. Results of this experiment were not good, however with a larger sample size results may improve.

K. *Cross validation*

This project involves using a dataset that is small compared to modern machine learning models. The size of the data was supposed to be representative of the number of images taken off a user's phone to spoof their noiseprint. However, to ensure that the test and training set have distributions that are representative of the other and to not overfit the model during training, we employ cross validation. Analyzing the results of the highest scoring model and our original score showed that there was not a significant improvement created by cross validation.

		Actual		
		cropped-iphone	cropped-galaxy8c	cropped-galaxy8l
Predicted	iphone	15	3	7
	galaxy8c	5	18	9
	galaxy8l	10	9	14

		Actual		
		denoised-iphone	denoised-galaxy8c	cropped-galaxy8l
Predicted	iphone	5	4	2
	galaxy8c	4	6	3
	galaxy8l	21	20	25

		Actual		
		in-iphone	in-galaxy8c	in-galaxy8l
Predicted	iphone	13	4	7
	galaxy8c	8	15	13
	galaxy8l	9	11	10

		Actual		
		8cn-iphone	8cn-galaxy8c	8cn-galaxy8l
Predicted	iphone	8	4	8
	galaxy8c	9	15	10
	galaxy8l	13	11	12

		Actual		
		8ln-iphone	8ln-galaxy8c	8ln-galaxy8l
Predicted	iphone	13	6	3
	galaxy8c	2	4	4
	galaxy8l	15	20	23

Table 8. Results of best performing model from cross validation

L. *2D-CNN model classification*

Since most recent models used for camera model classification are CNNs, a variety of experiments with CNNs as a classifier were also conducted. The preprocessing pipeline of the MLP converted a 2D image into a 1D input array. For the 2D-CNN, the input shape for the features were in 2 dimensions. Three different preprocessing modes were tested: raw images, noiseprints, LBP data. None of experiments used spoofed images in the testing set, only the capability for basic camera model classification was tested. Only two classes were used to test the 2D-CNN.

Raw images were tested as input as a test to see if the model could generate filters that automatically detect the noiseprint of the images. The experiment showed that currently the 2D-CNN is not able to classify to a high degree of accuracy. The preprocessing step that the MLP had contributed greatly towards the success of the model. While the training accuracy had an increasing trend, the validation accuracy revealed that the rise was just the model overfitting the training data.

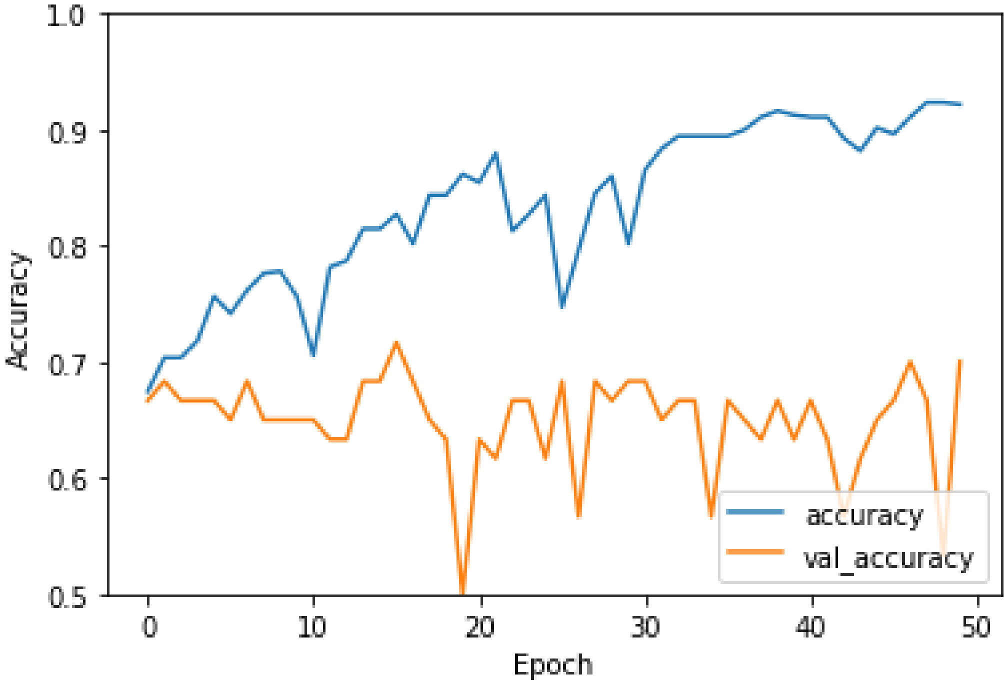

Figure 9. Accuracy vs Epochs for 2D-CNN with original images

Next, noiseprints were extracted from training images and used to train the 2D-CNN classifier. A variety of denoising methods were tested for the extraction of noiseprints of the training data. The best result is shown in Figure 10, however the validation accuracy was far below accuracy for the MLP classifier.

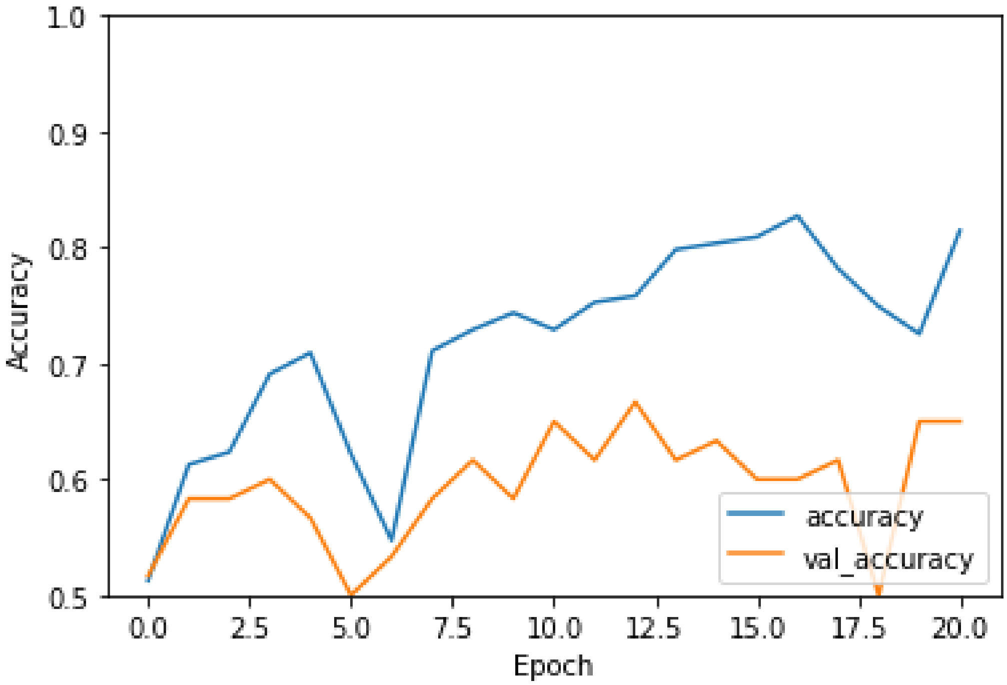

Figure 10. Accuracy vs Epochs for 2D-CNN with noiseprints

The last experiment with 2D-CNN's was to use LBP image data from the training images as input for the 2D-CNN. Each pixel in each training image was converted into a LBP number. The LBP array for each image was use for training. The accuracy was very poor. The differences in noiseprints were so small that even a CNN wasn't able to tell the difference between the two classes. These tests demonstrate the necessity and the strength of the preprocessing techniques used in the MLP model.

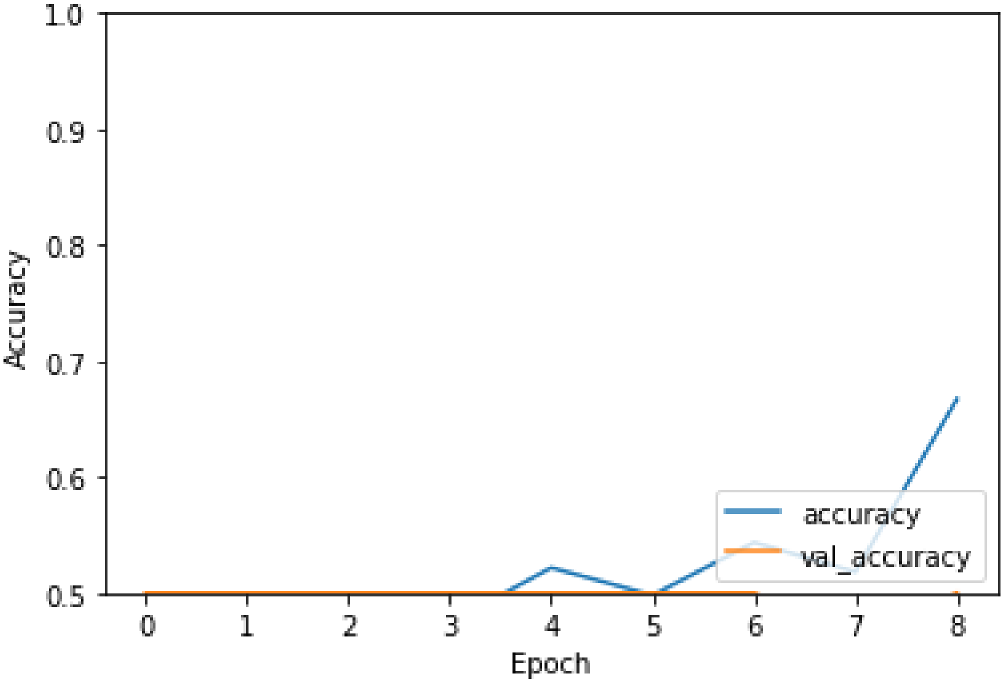

Figure 11. Accuracy vs Epochs for 2D-CNN with LBP data

M. *1D-CNN model classification*

With the desire to use the preprocessing techniques used in the MLP model and the convolution filters of a CNN, a 1D-CNN was created for this purpose. The same 159 length array used for input into the MLP model was fed into the 1D-CNN model. Very similar results were generated. While there was no improvement over the MLP, this potentially demonstrated that the behavior of a 1D-CNN was similar to an MLP for this type of classification.

		Actual	
		cropped-iphone	cropped-galaxy8c
Predicted	iphone	26	7
	galaxy8c	4	23

Actual

		denoised-iphone	denoised-galaxy8c
Predicted	iphone	18	19
	galaxy8c	12	11

Actual

		in-iphone	in-galaxy8c
Predicted	iphone	17	10
	galaxy8c	13	20

Actual

		gn-iphone	gn-galaxy8c
Predicted	iphone	17	12
	galaxy8c	13	18

Actual

		cropped-iphone	in-galaxy8c
Predicted	iphone	26	10
	galaxy8c	4	20

Actual

		cropped-galaxy8c	gn-iphone
Predicted	iphone	7	17
	galaxy8c	23	13

Table 9. Results of 1D-CNN classification

VII. Conclusion and Future Work

Neural Network models are a powerful tool towards the classification of camera models. However, numerous papers in addition to these experiments show that camera identification networks are not well equipped to deal with simple image augmentation-based attacks, and even less equipped to deal with adversarial based attacks. While not obtaining results that were expected coming into this experiment, I've determined that denoising methods play an integral role in the classification model that many papers overlook and standard measure of well performing denoising method should be established.

There are many areas for future work that build upon my research. A machine learning based method for PRNU extraction could be very powerful instead of a simple denoising filter. The most novel aspect of my research involves interclass model identification and with more powerful networks and denoising methods, high accuracy for this problem should be obtainable.

An AC-GAN could have also been used to train and spoof noiseprints. Using an AC-GAN would have reduced the complexity of the network of models need to spoof noiseprints. In my project, each camera model and noiseprint required a dedicated GAN to generate noiseprints. With an AC-GAN, there would have been only one model needed to spoof all types of camera models. It may have also generated noiseprints that fool the classifier better.

Advice towards attacking and fooling sensor noise classification models while maintaining high fidelity of the original image could be something as simple as denoising the image. These results show that denoised images, as well as random noise, reveal the models do have bias. A denoised image will have more coarsely patterned noise and classification models will likely pick one class more than another. Adding random noise at different degrees is a more effective way to skew the model to pick one class at a much larger rate. However, the SSIM of the noisy image and the original could reveal signs of a malicious attack that are obvious to the human eye.

Therefore, the level of noise injected should be discrete enough to avoid detection but large enough to make an impact on the network. Based on current tests, sensor noise captured is not very large and therefore is susceptible to attacks.

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] 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.

[3] Andrew Y. Ng and Michael I. Jordan. 2001. On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic* (*NIPS'01*). MIT Press, Cambridge, MA, USA, 841–848.

[4] Kurosawa, K., Kuroki, K., and Saitoh, N.: “CCD Fingerprint Method - Identification of a Video Camera from Videotaped Images,” Proc of ICIP' 99, Kobe, Japan, pp. 537–540, October 1999.

[5] D. Cozzolino, J. Thies, A. Rössler, M. Nießner and L. Verdoliva, "SpoC: Spoofing Camera Fingerprints," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021, pp. 990-1000, doi: 10.1109/CVPRW53098.2021.00110.

[6] C. Chen, X. Zhao and M. C. Stamm, "Generative Adversarial Attacks Against Deep-Learning-Based Camera Model Identification," in IEEE Transactions on Information Forensics and Security, doi: 10.1109/TIFS.2019.2945198.

[7] 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.

[8] A. Chambolle, “An algorithm for total variation minimization and applications, Journal of Mathematical Imaging and Vision”, Springer, 2004, 20, 89-97.

[9] کوردستان , "Kurdistan Nature, Landscape”. Openverse. https://creativecommons.org/licenses/by-sa/2.0/?ref=openverse

[10] s58y, "Canon FD 55mm f/1.2 S.S.C. Aspherical Lens (radioactive)". Openverse. https://creativecommons.org/licenses/by/2.0/?

[11] Webster, "Sony a7R III Digital Camera Sensor (29269149548)”. Openverse. https://creativecommons.org/licenses/by/2.0/?ref=openverse

[12] Soni, “Close up of a laptop motherboard”. Unsplash. https://unsplash.com/photos/Mo7RooYGXi4

[13] Xiawi, “Lbp computation”. Wikimedia.

https://commons.wikimedia.org/wiki/File:Lbp_computation.png

[14] Aphex34, “typical CNN architecture”. Wikimedia.

https://commons.wikimedia.org/wiki/File:Typical_cnn.png