University of California
Santa Barbara

Techniques for Visual Media Forensics

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Mike Goebel

December 2023

The Dissertation of Mike Goebel is approved.

Professor Kenneth Rose

Professor Shivkumar Chandrasekaran

Professor Ramtin Pedarsani

Professor B.S. Manjunath, Committee Chair

August, 2023

Techniques for Visual Media Forensics

Dedicated to my family and friends.

Acknowledgements

First, I would like to thank my advisor, B.S. Manjunath, for his continued support throughout my graduate studies. Professor Manjunath has always led with warmth and kindness, from the first day I visited UCSB as a perspective grad student in spring of 2018. His decades of knowledge in the field of computer vision truly broadened my perspective beyond the cookie-cutter deep learning teachings most students are entering the workforce with today. His guidance extended beyond the basics, teaching me how to make in impact through my PhD work.

I would also like to thank Kenneth Rose, Shivkumar Chandrasekaran, and Ramtin Pedarsani for being on my thesis committee, and offering insight throughout my degree. I have had the pleasure of taking classes with many of these faculty, broadening my knowledge vastly beyond my own area of research.

Next, I would like to acknowledge my colleagues at Mayachitra Inc. My numerous internships at the company were sprinkled throughout my tenure as a PhD student, and it led to many great collaborations. My work here not only elevated my skills as a researcher, but as an engineer delivering real-world products. This is especially true for all of my co-authors: Lakshmanan Nataraj, Tejaswi Nanjundaswamy, Tajuddin Manhar Mohammed, Shivkumar Chandrasekaran, Jason Bunk, and Chandrakanth Gudavalli.

Finally, all of my colleagues in the UCSB Vision Research Lab. I have made some of the best friends in my time here, and look forward to seeing many of you up in the Bay Area. I would first like to acknowledge my co-authors Satish Kumar, Abu Saleh Mohmmed Iftekhar Niloy, Devendra Jangid, Neal R Brodnik, Amil Khan, Po-Yu Kao, Angela Zhang, SaiSidharth Majeti, Raphael Ruschel Dos Santos, Jiaxiang (Tom) Jiang, and Ekta Prashnani. Second, other important people from the lab who have also had an impact on my time here: Fei Xu, Connor Levenson, Austin McEver, Anmol Kapoor,

Ivan Arevalo, Oytun Ulutan, Bowen Zhang,

Curriculum Vitæ
Mike Goebel

Education

September 2023	**Doctor of Philosophy** Electrical and Computer Engineering University of California, Santa Barbara, USA.
May 2018	**Master of Science** Electrical and Computer Engineering State University of New York at Binghamton.
May 2017	**Bachelor of Science** Electrical Engineering State University of New York at Binghamton.

Honors & Awards

2022	ECE Department Dissertation Fellowship, UCSB
2019	Outstanding TA in Electrical Engineering, UCSB
2017	Avangrid Foundation Scholarship, SUNY Binghamton

Publications

Michael Goebel, Jason Bunk, Srinjoy Chattopadhyay, Lakshmanan Nataraj, Shivkumar Chandrasekaran, and BS Manjunath. 2021. Attribution of gradient based adversarial attacks for reverse engineering of deceptions. arXiv preprint arXiv:2103.11002 (2021).

Michael Goebel, Arjuna Flenner, Lakshmanan Nataraj, and Bangalore S Manjunath. 2019. Deep learning methods for event verification and image repurposing detection. arXiv preprint arXiv:1902.04038 (2019).

Michael Goebel and BS Manjunath. 2020. Adversarial attacks on co-occurrence features for GAN detection. arXiv preprint arXiv:2009.07456 (2020).

Michael Goebel, Lakshmanan Nataraj, Tejaswi Nanjundaswamy, Tajuddin Manhar Mohammed, Shivkumar Chandrasekaran, and BS Manjunath. 2020. Detection, attribution and localization of gan generated images. arXiv preprint arXiv:2007.10466 (2020).

Chandrakanth Gudavalli, **Michael Goebel**, Tejaswi Nanjundaswamy, Lakshmanan Nataraj, Shivkumar Chandrasekaran, and BS Manjunath. 2023. Resampling Estimation based RPC Metadata Verification in Satellite Imagery. IEEE Transactions on Information Forensics and Security (2023).

Devendra K Jangid, Neal R Brodnik, **Michael G Goebel**, Amil Khan, SaiSidharth Majeti, McLean P Echlin, Samantha H Daly, Tresa M Pollock, and BS Manjunath. 2022.

Adaptable physics-based super-resolution for electron backscatter diffraction maps. npj Computational Materials 8, 1 (2022), 255.

Devendra K Jangid, Neal R Brodnik, Amil Khan, **Michael G Goebel**, McLean P Echlin, Tresa M Pollock, Samantha H Daly, and BS Manjunath. 2022. 3d grain shape generation in polycrystals using generative adversarial networks. Integrating Materials and Manufacturing Innovation 11, 1 (2022), 71–84.

Jiaxiang Jiang, Amil Khan, S Shailja, Samuel A Belteton, **Michael Goebel**, Daniel B Szymanski, and BS Manjunath. 2023. Segmentation, tracking, and sub-cellular feature extraction in 3D time-lapse images. Scientific Reports 13, 1 (2023), 3483.

Po-Yu Kao, Angela Zhang, **Michael Goebel**, Jefferson W Chen, and BS Manjunath. 2019. Predicting fluid intelligence of children using T1-weighted MR images and a StackNet. In Adolescent Brain Cognitive Development Neurocognitive Prediction: First Challenge, ABCD-NP 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 13, 2019, Proceedings 1, Springer International Publishing, 9–16.

Satish Kumar, ASM Iftekhar, **Michael Goebel**, Tom Bullock, Mary H MacLean, Michael B Miller, Tyler Santander, Barry Giesbrecht, Scott T Grafton, and BS Manjunath. 2021. StressNet: detecting stress in thermal videos. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 999–1009.

Lakshmanan Nataraj, **Michael Goebel**, Tajuddin Manhar Mohammed, Shivkumar Chandrasekaran, and BS Manjunath. 2021. Holistic image manipulation detection using pixel co-occurrence matrices. arXiv preprint arXiv:2104.05693 (2021).

Ekta Prashnani, **Michael Goebel**, and BS Manjunath. 2022. Generalizable Deepfake Detection with Phase-Based Motion Analysis. arXiv preprint arXiv:2211.09363 (2022).

Angela Zhang, S Shailja, Cezar Borba, Yishen Miao, **Michael Goebel**, Raphael Ruschel, Kerrianne Ryan, William Smith, and BS Manjunath. 2022. Automatic classification and neurotransmitter prediction of synapses in electron microscopy. Biological Imaging 2, (2022), e6.

Experience

12/2022-Present	Software Engineer, Google Ads, Alphabet Inc.
09/2018-12/2022	Graduate Student Researcher, Vision Research Lab, UCSB
06/2021-09/2021	Software Engineering Intern, Google Ads, Alphabet Inc.
06/2018-09/2020	Research Intern, Mayachitra Inc.

Programming Languages

Python, C++, SQL

Abstract

Techniques for Visual Media Forensics

by

Mike Goebel

Recent advances in image generation techniques have led to a proliferation of fake content online. This has created a need for adaptable, data driven methods to authenticate visual media, and prevent the spread of misinformation. In this defense, I will address two specific problem areas in detail.

First, the detection of GAN generated images. We demonstrate the efficacy of a novel forensic feature when combined with a conventional CNN architecture. Then, we investigate these in the adversarial setting, evaluating robustness to several levels of attack. These works represent some of the first attempts at adversarially robust GAN image detection.

The second area investigates methods for applying forensics to satellite images. Specifically, we look at the problems of metadata verification and image splicing detection. We demonstrate the power of incorporating satellite image metadata into the splicing detection problem, resulting in a joint model to accomplish both metadata and image verification with high accuracy.

The dissertation will conclude with a brief discussion of several other forensic areas, and potential future directions for the field.

Contents

Curriculum Vitae **vii**

Abstract **ix**

List of Figures **xii**

List of Tables **xv**

1 Introduction **1**

1.1 What is Visual Media Forensics? 1

1.2 Summary of Contributions 2

1.3 Organization 3

2 Background **4**

2.1 Understanding the Imaging Process 4

2.2 Overview of the Modern Image Forensics Field 7

2.3 Orthorectification 9

3 GAN Image Detection **20**

3.1 Overview of GAN Generation 21

3.2 Our GAN Detection Method 25

3.3 Datasets 29

3.4 Experiments 32

3.5 Adversarial Attack Setup 42

3.6 Adversarial Attack Methods 44

3.7 Adversarial Gray Box attack 45

3.8 Adversarial Attack Experiments 57

3.9 Conclusion 63

4 RPC Verification **64**

4.1 Background 64

4.2 Orthorectification and RPC Metadata 67

4.3 Method . . . 70
4.4 Experiments . . . 78
4.5 Conclusions . . . 82

5 RPC Splicing 83
5.1 Motivation . . . 83
5.2 Dataset . . . 84
5.3 State-of-the-Art Baseline . . . 90
5.4 RPC based Pixel Alteration Localization (RPC-PAL) Model . . . 91
5.5 Experimental Results . . . 97
5.6 RPC Splicing Detection Summary . . . 100

6 Discussions and Conclusion 103
6.1 DeepFake Detection . . . 104
6.2 Potential Future Directions . . . 106
6.3 Final Thoughts . . . 108

Bibliography 109

List of Figures

2.1 Visualization of the orthorectification process. The original and orthorectified images are on the top and bottom respectively. The red lines represent lines of equal latitude and longitude. 10
2.2 The satellite imaging process. From left to right: the camera location, imaging plane, and image field-of-view. 14
2.3 Parameters contained in the RPC metadata. 17

3.1 A basic GAN training setup 22
3.2 An example co-occurrence computation. The input image (a) is split into its three color channels (b). For each color channel, 4 different pairs of pixels are used to generate 2-dimensional histograms (c). Horizontal, vertical, diagonal, and anti-diagonal pairs are considered. These histograms are then stacked to produce a single tensor (d). For some tests, only a subset of the co-occurrence matrices will be used. 26
3.3 The original Xception network [1], shown next to our two modified models. Our architectures for detection and attribution are the same, except for the last layer and activation. 28
3.4 Sample images from different GAN datasets (a) Real images and (b) GAN images from different GAN datasets (top to bottom): ProGAN [2], StarGAN [3], CycleGAN [4], StyleGAN [5], and SPADE/GauGAN [6]. 29
(a) Real Images 29
(b) GAN Images 29
3.5 Quantitative summary of the GAN datasets used in our experiments. . . 30
3.6 Visualization of images from different GAN datasets using t-SNE [7]. . . 37
3.7 t-SNE visualization of 6 classes: Real, StyleGAN, StarGAN, ProGAN, SPADE/GauGAN and CycleGAN 40
3.8 Localization heatmaps of (a) Real images and (b) GAN images from different GAN datasets (top to bottom): ProGAN [2], StarGAN [3], CycleGAN [4], StyleGAN [5], and SPADE/GauGAN [6]. 41
(a) Real Images 41
(b) GAN Images 41

3.9 High-level diagram of the detection architectures. Sections 3.5.1, 3.5.2, and 3.5.3 describe the different feature extraction methods investigated. Section 3.8.2 describes the different CNN architectures tested. 43

3.10 Loss functions with respect to value in each index for 1D loss functions, on source of [1,2,3] and target of [2,3,4]. Ideally, we would like the gradients for each index, at initialization, to be negative. This will push each value in the source towards the target. L1 loss would often get stuck on the plataeus, L2 would get stuck at the minima in-between integer values, and the L1 pyramid converged more easily. See figure 3.11 for convergence results. 50

3.11 Plot of each value in the source vector with respect to step number, when solving with a source of [1,2,3] and target of [2,3,4]. Top left: without random noise, all points are stuck in flat regions, no change in values. Top right: the addition of noise allows for the algorithm using L_1 norm to gradually drift towards the target. Bottom row: Both with and without noise, the algorithm converges over 14 times faster than the top right case. 51

3.12 Example applying point-wise and pyramid loss to a 2D histogram gradient descent problem. Each graph is a parametric plot, with step number as the parameter, showing the path of each 2D input from source to solution. Each color represents a different 2D input element. Top left: with L_1 loss and no noise, all gradients are 0. Top right: With noise, L_1 finds a sub-optimal minima, where not all target points are reached. Bottom left: Without added noise, the pyramid loss almost converges to a global minima. However, the two source points originating from (1,1) need to split and fill different target points. With deterministic gradient descent, this splitting will not occur. Bottom right: A proper solution is found with pyramid loss and noise. 53

3.13 Example solution found by our algorithm, for the given real and GAN images. 54

3.14 Another example solution found by our algorithm, and corresponding red channel co-occurrence matrices. In the top-left corner of the solution co-occurrence a square artifact can be seen. 55

3.15 Testing of the co-occurrence algorithm on our dataset described in section 3.8.1 on 200 images for 3 different λ values. Smaller λ values will force the co-occurrence matrix of the adversarial image closer to that of the real image, at the cost of greater perturbation. 55

4.1 Visualization of Expected Residual DFT patterns when RPC Metadata is tampered (left) and Untampered (right), with respect to Residual DFT pattern of the image (middle). 66

4.2 Some example resampling patterns. Only the sub-integer shifts impact the noise variance patterns. (a) Pattern where noise variance is σ^2. (b) Pattern where noise variance is $0.25\sigma^2$. 68
4.3 Resampling spatial mapping for a scaling example (Scale factor of 0.9; blue: original locations; red: new locations). These sampling patterns move in and out of phase with each other to form a periodic pattern. This pattern is reflected in the noise variance of the resampled image. 69
4.4 Some toy examples showing the image Residual DFT (2nd row) under different transformations. 69
4.5 A grid before (left) orthorectification and after (right). Note the significant non-linearities introduced by the hilly region. 70
4.6 Overview of proposed framework to detect tampering in RPC Metadata. 73
4.7 Sample demonstration to show that SSIM is more apt than MSE for our use case. Even though both the DFT patterns look alike, MSE is unable to capture the similarity unlike SSIM. 77
4.8 Sample demonstration to show that SSIM is more apt than Cosine Similarity for our use case. Even though both the DFT patterns look alike, cosine similarity is unable to capture the similarity unlike SSIM. 78

5.1 A sample manipulated image and corresponding manipulation mask. . . . 85
5.2 The process of creating spliced images. Two images from the same geographic region are used. One provides the background, and the other, an object to be spliced in. Random scaling, rotation, and translation is applied. 87
5.3 A histogram of manipulation area in pixels for each region. Note the horizontal axis is a log scale. 89
5.4 Visualization of the different non-linearities. The soft absolute value function strikes a balance between the square function which is ideal under perfect noise estimation, and the Gaussian kernel, currently used by the previous resampling works. 95
5.5 Full RPC-PAL network diagram. The details of each computational step are described in the text. 96
5.6 ROC curves for the Japan test set. 98
5.7 ROC curves for the South America test set. 99
5.8 ROC curves for the West Africa test set. 99
5.9 Histogram of pixel detector responses for several scenarios. The RPC based model will flag most pixels which are spliced and/or have manipulated RPC metadata. 101

6.1 Visualization of the Complex Steerable Pyramid. The phase of the pyramid can be used to represent motion at different scales. 105

List of Tables

2.1 A sample set of RPC coefficients. The largest magnitude coefficients are in the linear terms, quickly shrinking in the higher order terms. 18

3.1 Comparison of different popular ImageNet [8] classification architectures on classifying GANs from co-occurrence matrices. All datasets are mixed for training, validation, and testing. The features are extracted from a whole image, with no JPEG compression. 33
3.2 Test on difference co-occurrence pairs. These were done on the whole image, with the additional challenge of JPEG compression. The JPEG quality factor was randomly selected with equal probability from the set of 75, 85, 90, or no JPEG compression 34
3.3 Accuracy when trained on one patch size, and tested on another. Data for training and testing has been pre-processed using JPEG compression with quality factors randomly selected from 75, 85, 90 or none. 34
3.4 Test accuracy when model is trained on images pre-processed with one JPEG quality factor, and tested on another. 35
3.5 Train on all but one GAN, test on the held out images. Patch size of 128, no JPEG compression. 36
3.6 Comparison with State-of-the-art. The co-occurrence based method outperforms all other methods in this evaluation metric. 36
3.7 Number of images per class 38
3.8 Confusion matrix on images from GAN datasets without any pre-processing on the full image. Equal prior accuracy of 0.9654. 40
3.9 Confusion matrix on images from GAN datasets without any pre-processing on 128×128 patches. Equal prior accuracy of 0.8477. 41
3.10 Confusion matrix with JPEG compression (128×128 patches). Equal prior accuracy of 0.8088. The images were preprocessed using a JPEG factor of 75, 85, 90, or no compression. Each of these four possible preprocessing functions was randomly selected with equal probability for every image. . 41

3.11 Combined number of real and fake samples from each data subset. For all subsets, the number of real and fake examples is equal. In all, there were 48k real and 48k GAN images used. 58
3.12 Overall accuracy of different networks on a balanced test set of real and GAN. Each row represents using a different deep learning architecture. Three detection methods are shown as the first column headers. The second shows results with either ImageNet weights or random initialization. 58
3.13 Test set accuracy of co-occurrence based detectors, without adversarial retraining. Gray-box (GB) co-occurrence (CO) samples are generated as described in section 3.7. PGD co-occurrence examples are produced using ResNet18. 59
3.14 Results for only the ResNet18 co-occurrence detector. The rows show results with and without adversarial retraining. 60
3.15 Comprehensive table showing test set results on all datasets, for many training combinations. Each row represents a different detector and/or training set, and each column the data set tested on. The PGD data is generated using the models labeled with an asterisk. 60
3.16 Results of cross-channel co-occurrence detector on cross-channel co-occurrence method described in section 3.8.5, without adversarial retraining. Adversarial images were generated using the gray-box method, using this new co-occurrence formulation. 62

4.1 Data collected from different regions of the globe to test the proposed method. 79
4.2 Performance for varying patch size 80
4.3 Experimental results on the datasets from different parts of globe. Each image stems two samples - one in the tampered dataset and the other in untampered dataset. 81

5.1 Total number of spliced images per country. 89
5.2 MantraNet Test Results after fine-tuning. 91
5.3 Comparison of all networks. Scores are reported as AUC-ROC. The combine RPC-PAL model out-performs all other models on each of the test sets. 97

6.1 PhaseForensics results for cross-dataset generalization, where our method achieves state of the art results. 106

Chapter 1

Introduction

1.1 What is Visual Media Forensics?

Given a shocking image or video, we may often ask ourselves if what we believe to be seeing is real. There exist many ”tricks” which can be used in deception. Illusions of perception have been explored long before the creation of the the first digital camera. UFOs, Bigfoot, and the Lock Ness Monster just to name a few.

But of specific interest to this work are the digital image and video manipulations. These manipulations present a lower barrier to entry in the modern world, requiring only a computer.

Unfortunately, the question of media authenticity is not black-and-white. Digital manipulations are often necessary to present the best picture possible for the situation. JPEG compression alters an image, for the benign purpose of decreasing the file size. Contrast enhancement, cropping, and zooming can all make a small area of interest more visible.

On the science side of things, a biology researcher may significantly adjust the saturation in a confocal microscope image, to show a particular sub-cellular structure. Application of this manipulation to all samples in a dataset may in fact create a more informative representation of the underlying biological structure. However, similar adjustment of levels in a satellite image used for detection of methane plumes may misrepresent the true patterns.

As another example, the same manipulation in a similar context may have drastically different implications for the truthfulness of an image. Modern software allows for everyday users to take a series of group photos in quick succession, and swap faces between each photo. If one person blinks in the first photo, their face from the second photo can realistically swapped from the second. However, the same technology may also be used to introduce new faces into the scene, creating a drastically different semantic meaning for the scene.

From the previous examples, it is clear that there can be no single method to authoritatively declare an image or video to be manipulated in all use cases. Instead, this work aims to build more pinpointed classifiers for specific tasks. This will provide users with more informative and accurate model responses, to accurately assess the media in the context they originally saw it.

1.2 Summary of Contributions

There are two primary areas of contribution in this thesis. First, in the realm of Generative Adversarial Network (GAN) image detection. This generation methodology was the leader in terms of quality for the majority of my thesis. In this work, we propose the multi-directional co-occurrence feature, combined with deep learning, as a method of detecting these images. We then do extensive analysis, demonstrating these networks ability for attribution of the type of GAN model used, as well as limits on the localization of manipulation. Finally for GAN detection, there is extensive work done in developing potential adversarial attacks against this feature. This work presents the first adversarial attack against the co-occurrence feature.

The next set of contributions pertain to the area of satellite image forensics. In one work, we identify the first method to verify the authenticity of satellite image RPC metadata using the image pixels. This is the first method to investigate the problem of satellite image RPC metadata authentication. This is also one of few works to investigate resampling detection and estimation in the case of non-linear transformations.

In another, we generalize this to include pixel tampering detection. This method is verified on a set of spliced satellite images. Our method significantly out-performs a state-of-the-art splicing localization deep learning method. It does so using a combination of the RPC metadata, and explainable signal processing methods.

1.3 Organization

- Chapter 2 provides background on the imaging pipeline, a summary image forensic techniques, and a description of the orthorectification process used in satellite imagery.
- Chapter 3 presents the work on GAN image detection.
- Chapter 4 describes the work on satellite image RPC metadata verification.
- Chapter 5 demonstrates the generalization of ideas in Chapter 4 to the new task of image splicing detection.
- Chapter 6 summarizes the main components of the dissertation, presents some of my other related work, and proposes new domains the core ideals of this work could be generalized to.

Chapter 2

Background

This chapter provides important background information for the remaining chapters. The first section describes a theoretical basis for image forensics. By describing the baseline principles under which an untampered image is generated, it becomes more apparent how deviations from this can be detected and used in forensics.

The second section provides background on general image forensics. This is meant to be a broad summary, and focuses mainly on forensics in handheld camera images.

The last section of this chapter described the details of the satellite imaging process. Especially relevant to this work is the process of orthorectification, where a perspective image from the satellite sensor is converted to ”map-like” orthogonal projection.

2.1 Understanding the Imaging Process

In theory, an image is simply an array of bytes. Without additional knowledge about the imaging process, there is no reason to say that one collection of bytes is more likely to be tampered than another. Understanding the imaging process can help identify expected patterns in the image pixels, and differentiate between tampered and untampered classes.

2.1.1 The pinhole camera model

The pinhole camera model represents one of the simplest mathematical models of an imaging system. In this model, all points in 3D space are projected through the pinhole, onto an imaging plane. The imaging plane would then be subdivided into a grid of pixels. To convert from the continuous image to the discrete image, an average of the continuous value can be taken inside of the pixel area.

This model is impossible to achieve fully in practice, but is intuitively what we might expect of an image. Then, specific deviations from this can be classified and analyzed individually.

2.1.2 Deviations from the Pinhole Camera Model

Several common deviations from the pinhole camera are reliably encountered with digital cameras.

Firstly, distortions are introduced by the lens. Compared to a simple pinhole, lenses allow for significantly more light to reach the sensor. However, they introduce blurring of objects which are too close or too far. Barrel, pincushion, and mustache distortion can also bring objects at the edge of the imaging frame closer or farther from the center. Finally, chromatic aberration can be present in multi spectral images. Just as a prism can separate the spectrum of white light, a lens can act differently on different spectrums of light. Given all of these potential lens distortions, and the high cost of manufacturing quality lenses, there is significant effort in many modern cameras to correct for these distortions in software. Such corrects may correct for the above distortions to a degree, but may themselves leave further artifacts in the image.

Another group of distortions arise from the discretization between a real camera, and the pinhole camera model. First, light itself arrives in discrete photons. Even in a perfectly fixed scene, multiple exposures will lead to different pixel outputs. However, it can be assumed that there is some underlying rate of photons, λ. The actual observed rate of photons will be a Poisson random variable, with an average rate of λ. Further noise is introduced as the finite photos pass through the color filter array, interact with the camera sensor, and eventually are read by the analog to digital converter into bits.

There exist several other readily modellable deviations - such as Bayer color filter array interpolation. There also exist more difficult to model deviations. For example, imperfections in the sensor manufacturing. [9] found that certain pixels within a sensor would be detectably more or less sensitive than others. This work then showed that these varying sensitivities produced a fingerprint which is unique to each individual sensor.

All together, the essence of what makes an image real is complex. There are likely intricacies about imaging we do not yet know about, or have represented in a model.

2.1.3 Image Forensics as a Statistical Problem

With the entire imaging process being non-deterministic, there are some interesting extremes worth considering in the context of forensics. First, imagine a setup taking a picture of a gray wall. With high probability, the image output will be gray, plus some amount of noise. In most practical imaging scenarios this signal to noise ratio will be small. However, there remains a non-zero probability that any possible combination of image pixel values could be produced.

Intuitively, we might think of the sets of manipulated and unmanipulated images being non-overlapping within a given context. Just as for object identification, cats and dogs are treated as two separate classes. However, this is not true for image forensics. Instead, it could be more accurately be thought of as the probability that a given image came from an authentic imaging process, compared to a process involving manipulation.

In many forensic works, there is a focus on the noise patterns. Here, noise will refer to deviations seen from an ideal pinhole camera model. Compared to noise, content is often complicated. It is much more difficult to have an accurate model for a 3D scene from a single picture than having a model for pixel noise. Noise in most cases can be accurately represented as random variables drawn from some distribution. This approach now poses forensics as a problem of identifying which distribution this noise is likely to arise from: real or manipulated.

2.2 Overview of the Modern Image Forensics Field

This section outlines several common image forensic tasks, and some of the state of the art methods to achieve them.

2.2.1 Common Image Forensics Tasks

Within image forensics, there are several related tasks. These are not mutually exclusive. However, thinking about the different objectives within forensics can give a better picture of the field.

Binary Classification on Images: Determine with a binary score, whether or not an image had undergone a particular manipulation. For example, GAN image detection (see Chapter 3).

Binary Classification on Metadata: Determine if the corresponding metadata matches the provided image. For example, RPC metadata verification (Chapter 4), or authentication of a camera model.

Parameter Estimation: Often instead of a binary score, we want to have more information about the image manipulation. Estimation of the scaling applied, or GAN generation method for an image are both examples.

Adversarial Forensics: Here, the party producing the manipulated image wishes not only to fool a human looking at an image, but also methods attempting to detect the manipulation. This area is investigated in Chapter 3.

Steganogaraphy and Watermarking: The task of hiding visually imperceivable data in an image. For steganography, the embedded data should be undetectable to anyone without a secret key. In watermarking, the watermark may be detectable by anyone, but is often designed to be difficult to remove. Both of these fields involve the embedding of such signals, and detection.

2.2.2 State of the Art Forensics Methods

Most state of the art methods fall into one of these classes:

- Training an off-the-shelf deep neural network directly.
- Hand crafting some feature based on theoretical assumptions, and passing this to a neural network.
- Designing a neural network architecture specifically tailored to a forensics task.

There is no single best state of the art model for all forensics tasks, as the tasks can vary quite significantly. Furthermore, there is often not agreement between datasets as to what is considered to be acceptable and unacceptable manipulations in terms of maintaining an image's authenticity.

However, some strategies have proven useful across tasks. The Bayar convolutional layers was one of those examples [10]. The novel approach in this paper was to constrain the first convolutional layer to have a -1 as the center weight value, and for all of the remaining weights to sum to zero. This effectively constrained the neural network to learn a high pass filter, by predicting the noise residual. Under the assumptions that patterns in the noise are useful for forensics, this constrained the network to only utilize what was considered useful by the network designer. The full convolutional neural network was used to classify images which may have a median filter, Gaussian blur, Gaussian noise, or resampling applied.

This same filter was used as a component in many other papers, including MantraNet [11]. MantraNet remains as one of the state of the art methods for image manipulation localization. It was trained on a broad set of over 300 manipulations, and created a novel covolutional block based on the z-score statistical test. Here, the assumption was that the features in manipulated regions would be anomalous relative to the other features in an image, and the z-score test could better constrain the network to detect this.

Other methods have taken advantage of more pinpointed artifacts which may show up in images. For example, the JPEG ghost artifacts first published by Farid et. al. [12].

Of particular interest to this thesis are the artifacts produced by orthorectification of satellite images. The details of this orthorectification is described in the next section. This serves as background information for the satellite image features used in chapters 4 and 5.

2.3 Orthorectification

Particular only to satellite images in this work, orthorectifaction is one additional process which can be in the imaging pipeline. The process is intricate, and plays a pivotal role in aiding in image forensic tasks in chapters 4 and 5.

The process of orthorectification results in a camera model which differs from the pinhole camera model commonly used in forensics. Instead, the camera projection is resampled into a different projection entirely. It is from this additional projection that artifacts can be uncovered for forensics.

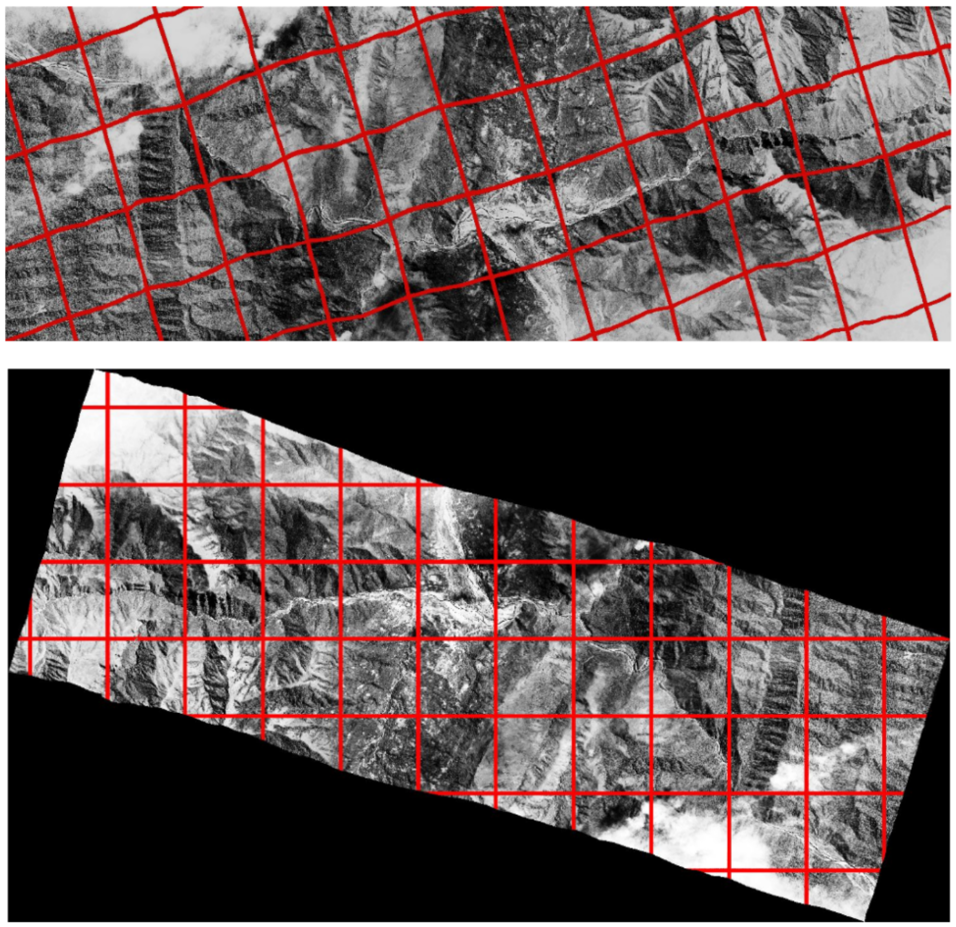


Figure 2.1: Visualization of the orthorectification process. The original and orthorectified images are on the top and bottom respectively. The red lines represent lines of equal latitude and longitude.

2.3.1 Orthorectification Definition

Intuitively, orthorectification is the process of warping a satellite image from a perspective image to a map-like one. In the orthorectified images, lines of equal latitude are equally spaced and horizontal. Line of equal longitude are also equally spaced, and vertical.

In the orthorectification process, there are three unique entities involved. These are:

- The perspective image: This is what is captured by the satellite sensor. It is a perspective image from the camera location.

- The Rational Polynomial Coefficients (RPCs): This captures the mapping from world coordinates (latitude, longitude, elevation) to camera pixel coordinates.
- Digital Elevation Maps (DEMs): A mapping from (latitude, longitude) to elevation, as a raster image.

To perform orthorectification, a mapping from (latitude, longitude) to (pixel row, pixel column) is needed. This mapping is achieved in a two step process. DEMs can be used to add elevation to the (latitude, longitude) pair. Then, (latitude, longitude, elevation) can be mapped to a pixel coordinate with the RPCs.

To preform an orthorectification, all three of the above items are given. Then, a grid of (latitude, longitude) pairs is constructed. Each coordinate in the grid is then mapped to a floating point pixel coordinate, a location in the observed image. Each point should fall between a set of 4 nearest neighbor integer pixel coordinates. Then, interpolation can be used to compute the pixel value at that lat-long location.

DEMs are not of particular interest to this work, as they are freely available online, and easier to verify than satellite image. As in, manipulation of a DEM could be easily checked against another elevation map. Ideally, there should be little deviation between sensors. Furthermore, there should be little temporal change in the DEMs. This is in contrast to visible spectrum satellite images, which are frequently used to demonstrate fast-changing events.

While the perspective image and DEMs are quite simple from their definition, the RPCs are less so. The role that RPCs fill is quite open-ended: simply create a mapping from pixel coordinates to earth coordinated. The next section discusses the structure of this transformation, and what is contained in the metadata.

2.3.2 Orthorectification in the Linear Case

This section walks through the derivation of the orthorectification equation under the most restrictive of assumptions. Once this baseline method of orthorectification is established, each idealized assumption can be broken down, and analyzed individually. The camera is assumed to follow the perfect pinhole model. Lines of latitude and longitude are equally spaced, and orthogonal. The earth is also assumed to be perfectly flat. The flat-earth assumption introduces surprisingly little deviation from the truth, for observed sample sizes. Over a typical imaging span of 20km, a flat plane will only deviate from the surface of a 12800km diameter sphere by 8m.

Coordinates on the earth can be represented as a set of three values, (latitude, longitude, elevation) $\rightarrow$ (x, y, z). The imaging plane coordinates, as rows and columns, are represented as the pair (r, c). Projection onto the imaging plane requires several mathematical steps. First, the establishment of a coordinate system for the imaging plane. The camera location is assumed to be the origin, and the imaging plane is a distance of 1 away from the origin. Transformation of coordinates in the p = (x, y, z) earth coordinates can be done using scaling, rotation, and a translation. Mathematically, this could be represented as

$$p_c = RSp_e + t$$

where

$p_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$, the camera coordinates

$p_e = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix}$, the earth coordinates

$$R = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \text{, a rotation matrix}$$

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \text{, a scaling matrix}$$

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \text{, a translation vector}$$

Multiplying out the R and S matrices, the p_c components can be expressed as

$$x_c = b_0 + b_1x_e + b_2y_e + b_3z_e$$

$$y_c = b_4 + b_5x_e + b_6y_e + b_7z_e$$

$$z_c = b_8 + b_9x_e + b_{10}y_e + b_{11}z_e$$

In other words, the transformation from 3D earth coordinates to 3D image coordinates is affine. Projection onto the imaging plane gives the equations:

$$r = \frac{y_c}{z_c} = \frac{b_4+b_5x_e+b_6y_e+b_7z_e}{b_8+b_9x_e+b_{10}y_e+b_{11}z_e}$$

$$c = \frac{x_c}{z_c} = \frac{b_0+b_1x_e+b_2y_e+b_3z_e}{b_8+b_9x_e+b_{10}y_e+b_{11}z_e}$$

This shows that if the earth were flat, the precise equation for this task would be Rational **Linear** Equations. However, latitude, longitude, and elevation do not form a Euclidean space. The earth curves, and lines of longitude become closer near the poles. This next section discusses generalization of this simple example to the full RPC equation.

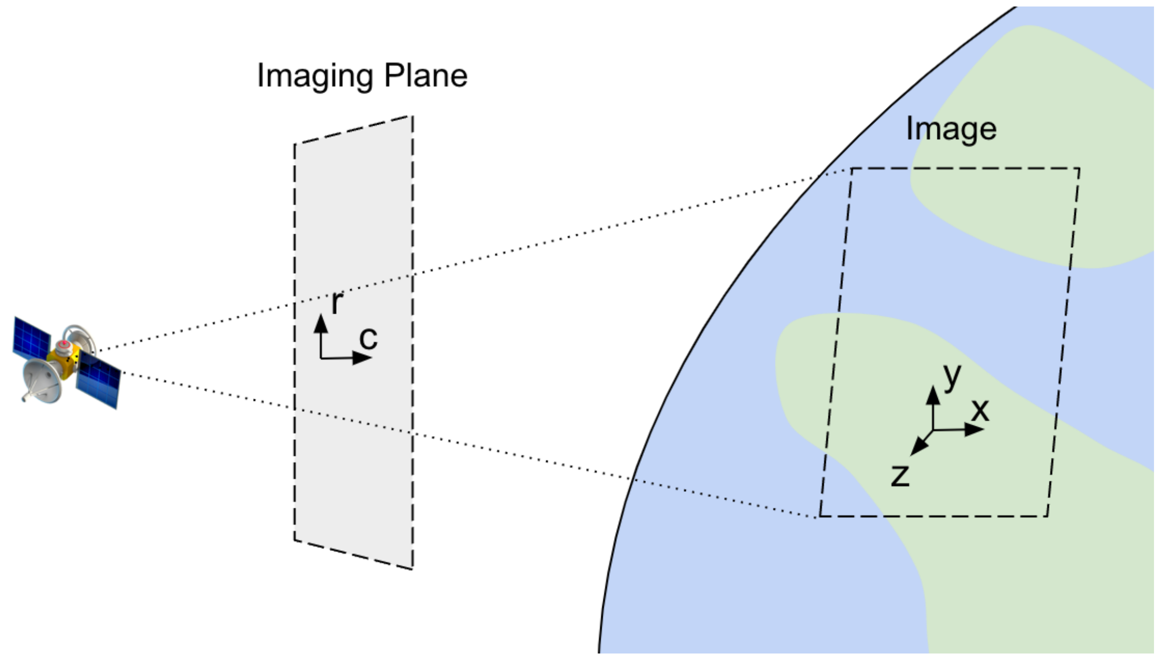


Figure 2.2: The satellite imaging process. From left to right: the camera location, imaging plane, and image field-of-view.

2.3.3 A more precise orthorectification equation

The assumption of a flat sea level over the imaging region allows for a simpler equation, but leaves an 8m error in the RPC rectification equation with respect to elevation. The real shape of the earth used by USGS is instead an oblate spheroid. The oblate spheroid model of the earth is rotationally symmetric about the north-south axis. Any crossections of the earth through the north-south axis are ellipses. The earth is slightly larger in the middle, with a flattening factor of about 1/298.

On the oblate spheroid earth model, locations can be defined by latitude (ϕ), longitude (θ), and elevation (d). These need to be mapped to Cartsian coordinates (x, y, z), where the origin is at the center of the earth sphere. The x-axis will travel through both the equator and prime meridian. The z-axis will travel through the north pole. And the y-axis will be orthogonal to both of these, traveling through the meridian of 90E. The conversion between these measurement systems is as follows:

$x = (d + R_1)cos(\phi)cos(\theta)$

$y = (d + R_1)cos(\phi)sin(\theta)$

$z = (d + R_2)sin(\phi)$

In practical applications, the ϕ and θ values within a particular image should only vary by a small amount. This would be assuming that the imaging is not done too close to the poles, where θ values from -180 to 180 could all be seen within a single image. However, all practical images will be far from the polar regions. These equations can be reparameterized based on this assumption. ϕ_0 and θ_0 are constants for an image, and could be considered as the average latitude and logitude in a region. ϕ_Δ and θ_Δ are variable within a region, but tend to be very small.

$x = (d + R_1)cos(\phi_0 + \phi_\Delta)sin(\theta_0 + \theta_\Delta)$

$y = (d + R_1)cos(\phi_0 + \phi_\Delta)cos(\theta_0 + \theta_\Delta)$

$z = (d + R_2)sin(\phi_0 + \phi_\Delta)$

Then, the following trig identities can be applied:

$sin(\alpha + \beta) = sin(\alpha)cos(\beta) + cos(\alpha)sin(\beta)$

$cos(\alpha + \beta) = cos(\alpha)cos(\beta) - sin(\alpha)sin(\beta)$

This yields new formulas for x, y, and z:

$x = (d + R_1)(cos(\phi_0)cos(\phi_\Delta) - sin(\phi_0)sin(\phi_\Delta))(sin(\theta_0)cos(\theta_\Delta) + cos(\theta_0)sin(\theta_\Delta))$

$y = (d + R_1)(cos(\phi_0)cos(\phi_\Delta) - sin(\phi_0)sin(\phi_\Delta))(cos(\theta_0)cos(\theta_\Delta) - cos(\theta_0)cos(\theta_\Delta))$

$z = (d + R_2)(cos(\phi_0)sin(\phi_\Delta) + sin(\phi_0)cos(\phi_\Delta))$

Taylor Series expansions can be constructed for $sin(\phi_\Delta)$, $cos(\phi_\Delta)$, $sin(\theta_\Delta)$, and $cos(\theta_\Delta)$. Given the very small ranges for θ_Δ and ϕ_Δ (on the order of 10^{-3} radians), the Taylor series can converge quite quickly.

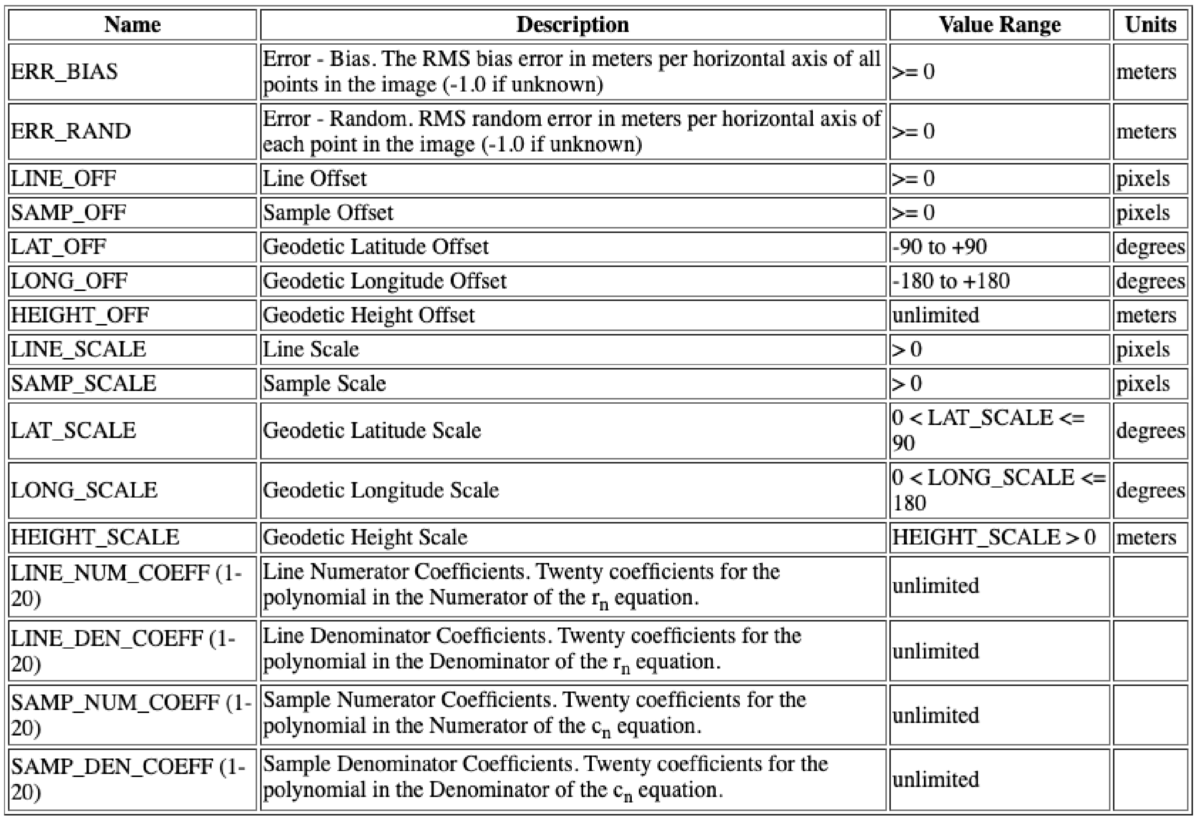
Here, we could re-visit the flat-earth approximation made in the linear case. If we approximate the same 20km segment of a 6400km radius circle with a parabola, a deviation of less than 1cm can be achieved. Further expansion beyond the second or third term in the polynomial provide close to no real benefit. For context, several tectonic plates on the earth move at a rate of several centimeters a year. Large earthquakes can even cause disturbances of up to 8 cm, almost instantaneously. For any practical system, regardless of how accurate the measurement is, there is almost no benefit to including higher order terms. However, there is additional computational benefit to truncating at 3rd order terms in the polynomials. Approximating this as a ratio of two polynomials reduces the need for trig functions in computation greatly. All of these figures are retrieved from the USGS WGS 1984 standard [13].

2.3.4 Definition of the RPC Metadata

The RPC equation consists of several distinct parts, beyond the rational polynomial coefficients. This includes the pre-scaling of the input and output values:

- An affine scaling of latitude, longitude, and height into the range [-1, 1]
- The rational polynomial transformation
- Another affine transformation, taking the row and column values from the range [-1, 1] to the pixel coordinates

$$\begin{aligned}
P &= (Latitude - LAT\_OFF)/LAT\_SCALE \\
L &= (Longitude - LONG\_OFF)/LONG\_SCALE \\
H &= (Elevation - ELEVATION\_OFF)/ELEVATION\_SCALE \\
R &= (Row - LINE\_OFF)/LINE\_SCALE \\
C &= (Column - SAMPLE\_OFF)/COLUMN\_SCALE
\end{aligned} \tag{2.1}$$



Name	Description	Value Range	Units
ERR_BIAS	Error - Bias. The RMS bias error in meters per horizontal axis of all points in the image (-1.0 if unknown)	>= 0	meters
ERR_RAND	Error - Random. RMS random error in meters per horizontal axis of each point in the image (-1.0 if unknown)	>= 0	meters
LINE_OFF	Line Offset	>= 0	pixels
SAMP_OFF	Sample Offset	>= 0	pixels
LAT_OFF	Geodetic Latitude Offset	-90 to +90	degrees
LONG_OFF	Geodetic Longitude Offset	-180 to +180	degrees
HEIGHT_OFF	Geodetic Height Offset	unlimited	meters
LINE_SCALE	Line Scale	> 0	pixels
SAMP_SCALE	Sample Scale	> 0	pixels
LAT_SCALE	Geodetic Latitude Scale	0 < LAT_SCALE <= 90	degrees
LONG_SCALE	Geodetic Longitude Scale	0 < LONG_SCALE <= 180	degrees
HEIGHT_SCALE	Geodetic Height Scale	HEIGHT_SCALE > 0	meters
LINE_NUM_COEFF (1-20)	Line Numerator Coefficients. Twenty coefficients for the polynomial in the Numerator of the r_n equation.	unlimited	
LINE_DEN_COEFF (1-20)	Line Denominator Coefficients. Twenty coefficients for the polynomial in the Denominator of the r_n equation.	unlimited	
SAMP_NUM_COEFF (1-20)	Sample Numerator Coefficients. Twenty coefficients for the polynomial in the Numerator of the c_n equation.	unlimited	
SAMP_DEN_COEFF (1-20)	Sample Denominator Coefficients. Twenty coefficients for the polynomial in the Denominator of the c_n equation.	unlimited	

Figure 2.3: Parameters contained in the RPC metadata.

Polynomial Term	Line Numerator	Line Denominator	Row Numerator	Row Denominator
1	0.0028087	1.0000000	-0.0018289	1.0000003
L	-1.0166277	-0.0000365	-0.0084539	0.0007958
P	-0.0027333	0.0000341	-1.1655187	-0.0005722
H	0.0011872	-0.0000018	-0.1105139	-0.0010652
$L \times P$	0.0066865	-0.0003074	-0.0409052	-0.0002819
$L \times H$	-0.0010402	-0.0002256	0.0018884	-0.0001988
$P \times H$	-0.0000099	-0.0000377	0.0004605	-0.0000098
L^2	-0.0082514	0.0012272	-0.0016286	-0.0006459
P^2	0.0000110	-0.0000634	0.0094496	-0.0002889
H^2	0.0000011	-0.0000000	-0.0000298	0.0000081
$P \times L \times H$	0.0000516	-0.0000002	0.0001124	0.0000013
L^3	-0.0012205	0.0000014	0.0002509	0.0000496
$L \times P^2$	0.0000298	-0.0000002	0.0007554	0.0000110
$L \times H^2$	-0.0000011	0.0000001	0.0000190	-0.0000003
$L^2 \times P$	0.0002998	-0.0000004	0.0003262	-0.0000010
P^3	0.0000001	-0.0000002	0.0002766	0.0000008
$P \times H^2$	-0.0000001	0.0000001	-0.0000071	0.0000002
$L^2 \times H$	0.0001795	-0.0000004	0.0002617	-0.0000004
$P^2 \times H$	0.0000001	-0.0000003	0.0000541	0.0000009
H^3	0.0000000	-0.0000000	-0.0000009	-0.0000000

Table 2.1: A sample set of RPC coefficients. The largest magnitude coefficients are in the linear terms, quickly shrinking in the higher order terms.

$$\begin{aligned} R &= \frac{\sum_{i=1}^{20} LINE\_NUM_i \times \rho(P,L,H)}{\sum_{i=1}^{20} LINE\_DEN_i \times \rho(P,L,H)} \\ C &= \frac{\sum_{i=1}^{20} LINE\_NUM_i \times \rho(P,L,H)}{\sum_{i=1}^{20} LINE\_DEN_i \times \rho(P,L,H)} \end{aligned} \tag{2.2}$$

2.3.5 RPC Metadata Summary

The RPC metadata, combined with the DEMs, defines a function to warp a perspective image to an orthorectified one. This warping is leveraged in Chapters 4 and 5 for forensics purposes.

The next chapter presents work exclusively focused on GAN detection. It will leverage aspects of the imaging pipeline defined in the first part of this chapter to help identify differences between real and GAN images.

Chapter 3

GAN Image Detection

This chapter presents a combination of work from several papers. First, some work focusing on exclusively on non-adversarial GAN image detection, attribution, and localization [14]. This portion discusses, in depth, the various tasks surrounding detection. The corresponding paper was one of the earliest successful works in detection, attribution, and localization of GANs. It was also one of the first to consider co-occurrence matrices for such a forensics task. This work was completed in collaboration with Lakshmanan Nataraj. Specifically, the methods development was done jointly, followed by test and evaluation and experimentation that I was mostly responsible for.

The second work discusses several methods for evading these detectors [15]. Here, the data used in the previous publication was reused for the adversarial work. This publication was the first to incorporate co-occurrence matrices into an adversarial framework. While the focus here is on GAN images, the findings could be generalized to other use cases of co-occurrence matrices.

3.1 Overview of GAN Generation

The advent of Convolutional Neural Networks (CNNs) [16, 17] has shown application in a wide variety of image processing tasks, and image manipulation is no exception. In particular, Generative Adversarial Networks (GANs) [18] have been one of the most promising advancements in image enhancement and manipulation. Due to the success of using GANs for image editing, it is now possible to use a combination of GANs and off-the-shelf image-editing tools to modify digital images to such an extent that it has become difficult to distinguish doctored images from normal ones.

The GAN training procedure involves a generator and discriminator, as shown in Figure 3.1. The generator may take in an input image and a desired attribute to change, then output an image containing that attribute. The discriminator will then try to differentiate between images produced by the generator and the authentic training examples. The generator and discriminator are trained in an alternate fashion, each attempting to optimize its performance against the other. Ideally, the generator will converge to a point where the output images are so similar to the ground truth that a human will not be able to distinguish the two. In this way, GANs have been used to produce “fake” images that are very close to the real input images.

These include image-to-image attribute transfer (CycleGAN [4]), generation of facial attributes and expressions (StarGAN [3]), as well as generation of whole new images such as faces (ProGAN [2], StyleGAN [5]), indoors (StyleGAN) and landscapes (SPADE/GauGAN [6]). In digital image forensics, the objective is to both detect these fake GAN generated images, localize areas in an image which have been generated by GANs, as well as identify which type of GAN was used in generating the fake image.

In the GAN training setup, the discriminator functions directly as a classifier of GAN and non-GAN images. So the question could be raised as to *why not use the GAN*

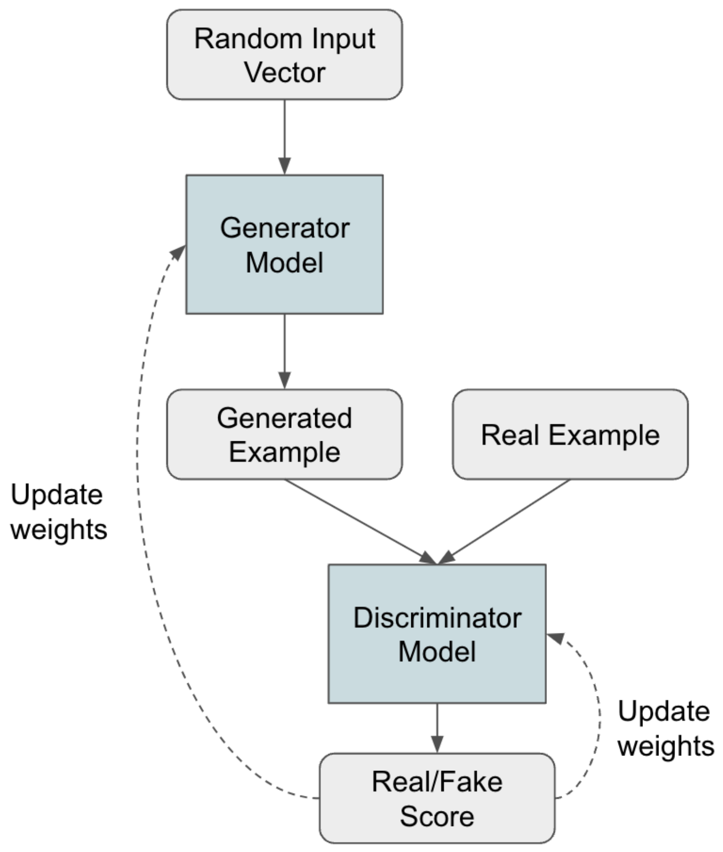


Figure 3.1: A basic GAN training setup

discriminator to detect if it's real or fake? To investigate this, we performed a quick test using the CycleGAN algorithm under the maps-to-satellite-images category, where fake maps are generated from real satellite images, and vice versa. In our test, we observed that the discriminator accuracy over the last 50 epochs was only 80.4%. However, state-of-the-art deep learning detectors for CycleGAN often achieve over 99% when tested on the same type of data which they are trained [19–21]. Though the discriminator fills its role of producing a good generator, it does not compare performance wise to other methods which have been suggested for detection.

While the visual results generated by GANs are promising, the GAN based techniques alter the statistics of pixels in the images that they generate. Hence, methods that look for deviations from natural image statistics could be effective in detecting GAN generated fake images. These methods have been well studied in the field of steganalysis which aims to detect the presence of hidden data in digital images. One such method is based on analyzing co-occurrences of pixels by computing a co-occurrence matrix. Traditionally, this method uses hand crafted features computed on the co-occurrence matrix and a machine learning classifier such as support vector machines determines if a message is hidden in the image [22, 23]. Other techniques involve calculating image residuals or passing the image through different filters before computing the co-occurrence matrix [24–26]. Inspired by steganalysis and natural image statistics, we propose a novel method to identify GAN generated images using a combination of pixel co-occurrence matrices and deep learning. In this proposed method, the co-occurrence matrix is used as an input to the deep learning model. In this chapter, we consider three related tasks. The first is detection, which aims to assign a binary real/fake score to each image. The second is attribution, which assigns a classification as to which GAN architecture produced the image in question. And the third is localization, which identifies which regions of an image may be GAN generated. For detection, we consider a two class framework - real and GAN, where a network is trained on co-occurrence matrices computed on the whole image to detect if an image is real or GAN generated. For attribution, the same network is trained in a multi-class setting depending on which GAN the image was generated from. For localization, a network is trained on co-occurrence matrices computed on image patches and a heatmap was is generated to indicate which patches are GAN generated. Detailed experimental results on large scale GAN datasets comprising over 2.76 million images originating from multiple diverse and challenging datasets generated using GAN based methods show that our approach is promising and will be an effective method for tackling future challenges of GANs.

3.1.1 GAN related works

Since the seminal work on GANs [18], there have been several hundreds of papers on using GANs to generate images. These works focus on generating images of high perceptual quality [2, 27–32], image-to-image translations [4, 30, 33], domain transfer [34, 35], super-resolution [36], image synthesis and completion [37–39], and generation of facial attributes and expressions [3, 35, 40, 41]. Several methods have been proposed in the area of image forensics over the past years [42–46]. Recent approaches have focused on applying deep learning based methods to detect tampered images [10, 26, 47–51].

In digital image forensics, detection of GAN generated images has been an active topic in recent times and several papers have been published in the last few years [19–21, 52–73, 73–76]. Other similar research include detection of computer generated images (CGI) [77–80]

In [19], Marra et al. compare various methods to identify CycleGAN images from normal ones. The top results they obtained are using a combination of residual features [26, 81] and deep learning [1]. In [53], Li et al. compute the residuals of high pass filtered images and then extract co-occurrence matrices on these residuals, which are then concatenated to form a feature vector that can distinguish real from fake GAN images. In [21], Zhang et al. identify an artifact caused by the up-sampling component included in the common GAN pipeline and show that such artifacts are manifested as replications of spectra in the frequency domain and thus propose a classifier model based on the spectrum input, rather than the pixel input. Our work is in some ways similar to [21]. However, instead of the DFT, we utilize the co-occurrence matrix.

3.2 Our GAN Detection Method

The method consists of two main steps. First the computation of a co-occurrence feature to capture the texture signatures of an image. The second involves passign this feature through a deep learning network. Each of these are described in the following two sections.

3.2.1 Co-Occurrence Matrix Computation

The co-occurrence matrices represent a two-dimensional histogram of pixel pair values in a region of interest. The vertical axis of the histogram represents the first value of the pair, and the horizontal axis, the second value. Equation 3.1 shows an example of this computation for a vertical pair.

$$C_{i,j} = \sum_{m,n} \begin{cases} 1, & I[m,n] = i \, and \, I[m+1,n] = j \\ 0, & otherwise \end{cases} \tag{3.1}$$

Under the assumption of 8-bit pixel depth, this will always produce a co-occurrence matrix of size 256x256. This is a key advantage of such a method, as it will allow for the

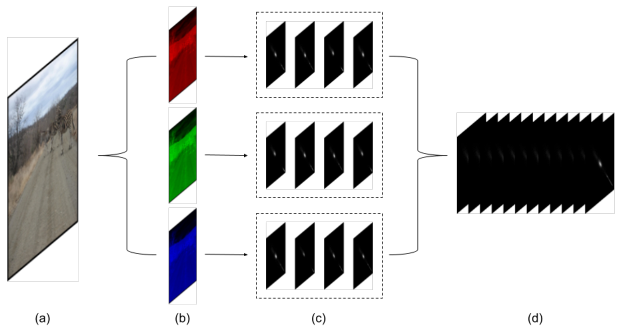


Figure 3.2: An example co-occurrence computation. The input image (a) is split into its three color channels (b). For each color channel, 4 different pairs of pixels are used to generate 2-dimensional histograms (c). Horizontal, vertical, diagonal, and anti-diagonal pairs are considered. These histograms are then stacked to produce a single tensor (d). For some tests, only a subset of the co-occurrence matrices will be used.

same network to be trained and tested on a variety of images without resizing.

Which pairs of pixels to take was one parameter of interest in our tests. For any pixel not touching an edge, there are 8 possible neighbors. We consider only 4 of these for our tests; right, bottom right, bottom, and bottom left. The other 4 possible pairs will provide redundant information. For example, the left pairs are equivalent to swapping the order of the first and second pixel in the right pair. In the co-occurrence matrix, this corresponds to a simple transpose. There are many subsets of these 4 pairs which could be taken, but our tests consider only a few; horizontal, vertical, horizontal and vertical, or all.

Before processing matrices through a CNN, some pre-processing is done. First, each co-occurrence matrix is divided by its maximum value. Given that the input images may be of varying sizes, this will force all inputs into a consistent scale. After normalization, all co-occurrence matrices for an image are stacked in the depth dimension. In the example of an RGB image with all 4 co-occurrence pairs, this will produce a new image-like feature tensor of size 256x256x12. Figure 3.2 gives a visualization of this process.

3.2.2 Backbone Convolutional Neural Networks

While the co-occurrence matrices are not themselves images, treating them as so has some theoretical backing. One of the primary motivations for using CNNs in image processing is their translation invariance property. In the case of a co-occurrence matrix, a translation along the main diagonal corresponds to adding a constant value to the image. We would not expect this manipulation to affect the forensic properties.

In this chapter, we use Xception Net [1] deep neural network architecture for detection, attribution and localization of GAN generated images. The Xception network is a modified version of Inception network [82] but was created under a stronger theoretical assumption than the original Inception, where cross-channel correlations are completely split from spatial correlations by use depth-wise separable convolutions. The network also includes residual connections, as shown in Figure 3.3. For these reasons, the authors claim that Xception can more easily find a better convergence point than most other CNN architectures, while keeping model capacity low [1]. In this paper, we modify the original input and output shapes in the Xception network to accommodate our task as shown in Figure 3.3. The initial convolutional portions of the network remain unchanged, though the output sizes of each block are slightly different. This small change in size is accommodated by the global pooling step. Finally, the last fully connected layer of each network is changed to the desired number of output classes, and given the appropriate activation. For detection and attribution, our architectures are the same except for the last layer and activation. For localization, no changes were made to the model architecture but co-occurrence matrices were extracted on small image patches, and individually passed through the network.

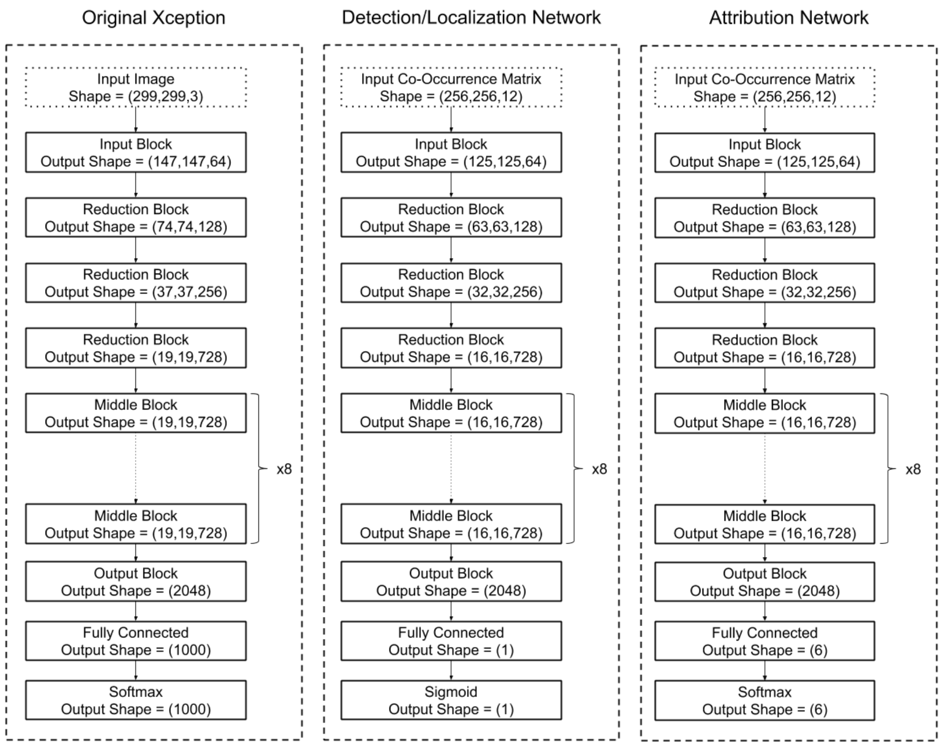
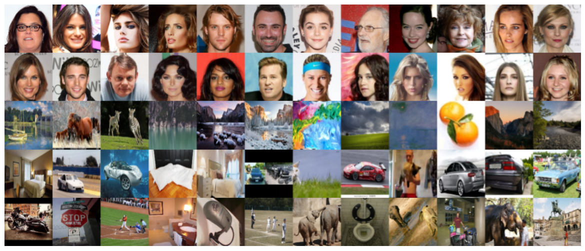

Original Xception
Detection/Localization Network
Attribution Network
Input Image Shape = (299,299,3)
Input Block Output Shape = (147,147,64)
Reduction Block Output Shape = (74,74,128)
Reduction Block Output Shape = (37,37,256)
Reduction Block Output Shape = (19,19,728)
Middle Block Output Shape = (19,19,728)
Middle Block Output Shape = (19,19,728)
x8
Output Block Output Shape = (2048)
Fully Connected Output Shape = (1000)
Softmax Output Shape = (1000)
Input Co-Occurrence Matrix Shape = (256,256,12)
Input Block Output Shape = (125,125,64)
Reduction Block Output Shape = (63,63,128)
Reduction Block Output Shape = (32,32,256)
Reduction Block Output Shape = (16,16,728)
Middle Block Output Shape = (16,16,728)
Middle Block Output Shape = (16,16,728)
x8
Output Block Output Shape = (2048)
Fully Connected Output Shape = (1)
Sigmoid Output Shape = (1)
Input Co-Occurrence Matrix Shape = (256,256,12)
Input Block Output Shape = (125,125,64)
Reduction Block Output Shape = (63,63,128)
Reduction Block Output Shape = (32,32,256)
Reduction Block Output Shape = (16,16,728)
Middle Block Output Shape = (16,16,728)
Middle Block Output Shape = (16,16,728)
x8
Output Block Output Shape = (2048)
Fully Connected Output Shape = (6)
Softmax Output Shape = (6)

Figure 3.3: The original Xception network [1], shown next to our two modified models. Our architectures for detection and attribution are the same, except for the last layer and activation.


(a) Real Images

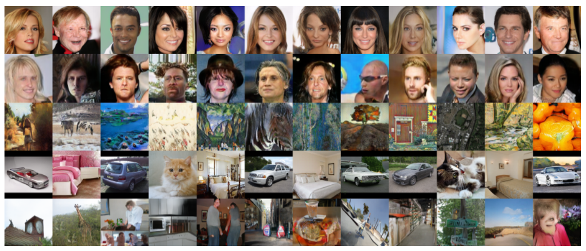

(b) GAN Images

Figure 3.4: Sample images from different GAN datasets (a) Real images and (b) GAN images from different GAN datasets (top to bottom): ProGAN [2], StarGAN [3], CycleGAN [4], StyleGAN [5], and SPADE/GauGAN [6].

3.3 Datasets

We evaluated our method on five different GAN architectures, of which each was trained on several different image generation tasks: ProGAN [2], StarGAN [3], CycleGAN [4], StyleGAN [5], and SPADE/GauGAN [6]. The modifications included image-to-image translation, facial attribute modification, style transfer, and pixel-wise semantic label to image generation. A summary of the datasets, including the number of images from each class, is shown in Figure 3.5. These comprise a total of more than 2.76 million images of which 1.69 million images are real images and 1.07 million images are fake GAN generated images. In several cases, one or more images in the GAN generated category will be directly associated with an image in the authentic class. For example, a person's headshot untampered, blond, aged, and gender reversed will all be in the dataset. However, the splitting for training accounts for this, and will keep all of these images together to be put into either training, validation, or test. Some sample images from all the GAN datasets are shown in Figure 3.4.

```
                          real    fake
base                   1696998 1073582
├── stargan               3279   29511
│   └── celeba            3279   29511
│
├── cyclegan             18151   18151
│   ├── map2sat           1096    1096
│   ├── ukiyoe            1500    1500
│   ├── vangogh           1500    1500
│   ├── horse2zebra       2401    2401
│   ├── cezanne           1500    1500
│   ├── cityscapes        2975    2975
│   ├── apple2orange      2014    2014
│   ├── summer2winter     2193    2193
│   ├── monet             2572    2572
│   └── facades            400     400
│
├── progan               30000   74000
│   └── celeba_hq        30000   74000
│
├── spade               145497  145497
│   ├── ade20k           22210   22210
│   └── coco_stuff      123287  123287
│
└── stylegan           1500071  806423
    ├── bedroom_lsun    500000  278790
    ├── cat_lsun        500071  279867
    └── car_lsun        500000  247766
```

Figure 3.5: Quantitative summary of the GAN datasets used in our experiments.

StarGAN

This dataset consists of only celebrity photographs from the CelebA dataset [83], and their GAN generated counterparts [3]. The GAN changes attributes of the person to give them black hair, brown hair, blond hair, different gender, different age, different hair and gender, different hair and age, different gender and age, or different hair, age, and gender. These are the smallest of all of the training images, being a square of size 128 pixels.

CycleGAN

This datasets includes image-to-image translations between a wide array of image classes [4]. The sets horse2zebra, apple2orange, and summer2winter do a strict image-to-image translation, with the assumption that the GAN will learn the areas to modify. While the whole output is generated by the GAN, the changes for these will ideally be more localized. Ukiyoe, Vangogh, Cezanne, and Monet are four artists which the GAN attempts to learn a translation from photographs to their respective styles of painting. Facades and cityscapes represent the reverse of the image segmentation task. Given a segmentation map as input, they produce an image of a facade or cityscape. Map2sat takes in a Google Maps image containing road, building, and water outlines, and generates a hypothetical satellite image.

ProGAN

This dataset consists of images of celebrities, and their GAN generated counterparts, at a square size of 1024 pixels [2]. All data was obtained per the instructions provided in the paper's Github repository.

SPADE/GauGAN

SPADE/GauGAN contains realistic natural images generated using GANs [6]. This dataset uses images from ADE20k [84] dataset containing natural scenes and COCO-Stuff [85] dataset comprising day-to-day images of things and other stuff, along with their associated segmentation maps. These untampered images are considered as real images in the GAN framework, and the pretrained models provided by the SPADE/GauGAN authors are used to generate GAN images from the segmentation maps.

StyleGAN

This dataset contains realistic images of persons, cars, cats and indoor scenes [5]. Images for this dataset were provided by the authors.

3.4 Experiments

This section describes the training procedure, and numerous tests done with the model. The tests involve tasks of detection, attribution, and localization. Testing also investigates several real-world problems, such as the effect of JPEG compression and generalization to unseen networks.

3.4.1 Training Procedure

All deep learning experiments in this paper were done using Keras 2.2.5 and all training was done using an Adam optimizer [86], learning rate of 10^{-4}, and cross-entropy loss. A batch size of 64 was used for all experiments. Unless otherwise stated, a split of 90% training, 5% validation, and 5% test was used. Given the large amount of data available, a single iteration through the entire dataset for training took 10 hours on a single Titan RTX GPU. To allow for more frequent evaluation on the validation set, the length of an epoch was capped at 100 batches. Validation steps were also capped at 50 batches, and test sets at 2000 batches. After training for a sufficient period of time for the network to converge, the checkpoint which scored the highest in validation was chosen for testing. For experiments to determine hyper-parameters, training was capped at 50 epochs, and took approximately 3 hours each on a single Titan RTX. After determination of hyper-parameters, training of the final model was done for 200 epochs, taking approximately 12 hours.

3.4.2 Comparison with other CNN architectures:

First we evaluate our method on different well known CNN architectures: VGG16 [17], ResNet50 ResNet101 [87], ResNet50V2, ResNet101V2 and ResNet152V2 [88], InceptionV3 and InceptionResNetV2 [82], and Xception [1]. Shown in Table 3.1 are the results

Table 3.1: Comparison of different popular ImageNet [8] classification architectures on classifying GANs from co-occurrence matrices. All datasets are mixed for training, validation, and testing. The features are extracted from a whole image, with no JPEG compression.

Network	Accuracy
VGG16 [17]	0.6115
ResNet50 [87]	0.9677
ResNet101 [87]	0.9755
ResNet152V2 [88]	0.9795
ResNet50V2 [88]	0.9856
InceptionResNetV2 [82]	0.9885
InceptionV3 [89]	0.9894
ResNet101V2 [88]	0.9900
Xception [1]	**0.9916**

for the different CNN networks. Though designed for ImageNet classification, all models take in an image with height, width, and 3 channels, and output a one-hot encoded label. The models are used as-is, with the following slight modifications. First, the number of input channels is set to be the depth of the co-occurrence feature tensor. Second, input shape was fixed at 256x256. Third, the number of output channels was set to 1. All of these parameters were passed as arguments to the respective Keras call for each model. A small margin separated the top performers, though Xception was the best with an accuracy of **0.9916** and had fewer parameters than others. For this reason, we chose Xception for the remainder of the experiments.

3.4.3 Comparison of Co-occurrence Matrix Pairs

Next we perform tests with different co-occurrence pairs, shown in Table 3.2. These experiments included JPEG compression, randomly selected from quality factors of 75, 85, 90, and no compression. Interestingly, it seems that the addition of more co-occurrence pairs did not significantly improve performance. For the remainder of the test, all 4 co-occurrence pairs were used.

Table 3.2: Test on difference co-occurrence pairs. These were done on the whole image, with the additional challenge of JPEG compression. The JPEG quality factor was randomly selected with equal probability from the set of 75, 85, 90, or no JPEG compression

Pairs	Accuracy
Horizontal	95.51
Vertical	95.56
Hor and Ver	95.17
Hor, Ver, and Diag	**95.68**

Table 3.3: Accuracy when trained on one patch size, and tested on another. Data for training and testing has been pre-processed using JPEG compression with quality factors randomly selected from 75, 85, 90 or none.

		Train		
		64	128	256
Test	64	0.7814	0.7555	0.6778
	128	0.8273	0.8336	0.8158
	256	0.8311	0.8546	0.8922

3.4.4 Effect of patch size

For real world applications, the two parameters of interest were JPEG compression and patch size. The results for different patch sizes are shown in Table 3.3. These results are from images JPEG compressed by a factor randomly selected from 75, 85, 90, and none. A model is trained for each of the possible patch sizes, and then each model is tested against features from each patch size. It should be noted that in cases where the input image is smaller than the requested patch size, the whole image is used. There is notable generalization between different patch sizes, in that the model trained on a patch size of 256 and tested on 128 achieves an accuracy within a few percentage points of a model trained and tested on 128. Thus we would expect our models to work with a variety of untested patch sizes within a reasonable range while only taking a minor performance drop.

Table 3.4: Test accuracy when model is trained on images pre-processed with one JPEG quality factor, and tested on another.

		Train			
		75	85	90	None
Test	75	0.7738	0.7448	0.7101	0.6605
	85	0.8209	0.8593	0.8362	0.7209
	90	0.8310	0.8690	0.8756	0.7651
	None	0.9198	0.9386	0.9416	0.9702

3.4.5 Effect of JPEG compression

Now assuming a fixed patch size of 128, we varied the JPEG quality factors: 75, 85, 90 and no compression. The model was again trained only on one particular JPEG factor as shown in Table 3.4. As expected, we see that performance increases with respect to quality factor. However, this table also shows that the model does not overfit to a particular quality factor, in that testing on a slightly better or worse quality factor gives a score not far from a model tuned to the particular test quality factor.

3.4.6 Generalization

To test the generalization between GANs, leave-one-out cross validation was used for each GAN architecture. One dataset of GAN images is used for testing and remaining GAN image datasets are used for training. Here, a patch size of 128 was used with no JPEG compression. From Table 3.5, we see that some GAN datasets such as SPADE, StarGAN and StyleGAN have high accuracy and are more generalizable. However, the accuracies for CycleGAN and ProGAN are lower in comparison, thus suggesting that images from these GAN categories should not be discarded when building a bigger GAN detection framework.

Visualization using t-SNE: To further investigate the variability in the GAN detection accuracies under the leave-one-out setting, we use t-SNE visualization [7] from outputs of the penultimate layer of the CNN, using images from the test set (as shown in Figure 3.6).

Table 3.5: Train on all but one GAN, test on the held out images. Patch size of 128, no JPEG compression.

Test GAN	Accuracy
StarGAN	0.8490
CycleGAN	0.7411
ProGAN	0.6768
SPADE	0.9874
StyleGAN	0.8265

Table 3.6: Comparison with State-of-the-art. The co-occurrence based method out--performs all other methods in this evaluation metric.

Method	ap2or	ho2zeb	wint2sum	citysc.	facades	map2sat	Ukiyoe	Van Gogh	Cezanne	Monet	Average
Steganalysis feat.	0.9893	0.9844	0.6623	1.0000	0.9738	0.8809	0.9793	0.9973	0.9983	0.9852	0.9440
Cozzalino2017	0.9990	0.9998	0.6122	0.9992	0.9725	0.9959	1.0000	0.9993	1.0000	0.9916	0.9507
XceptionNet	0.9591	0.9916	0.7674	1.0000	0.9856	0.7679	1.0000	0.9993	1.0000	0.9510	0.9449
Nataraj2019	0.9978	0.9975	0.9972	0.9200	0.8063	0.9751	0.9963	1.0000	0.9963	0.9916	0.9784
Zhang2019	0.9830	0.9840	0.9990	1.0000	1.0000	0.7860	0.9990	0.9750	0.9920	0.9970	0.9720
Proposed approach	0.9982	0.9979	0.9982	0.9366	0.9498	0.9776	0.9973	0.9980	0.9993	0.9697	**0.9817**

The t-SNE algorithm aims to reduce dimensionality of a set of vectors while preserving relative distances as closely as possible. While there are many solutions to this problem for different distance metrics and optimization methods, KL divergence on the Student-t distribution used in t-SNE has shown the most promising results on real-world data [7].

To limit computation time, no more than 1000 images were used for a particular GAN from either the authentic or GAN classes. As recommended in the original t-SNE publication, the vector was first reduced using Principle Component Analysis (PCA). The original 2048 were reduced to 50 using PCA, and passed to the t-SNE algorithm. As we see in Figure 3.6, the images in CycleGAN and ProGAN are more tightly clustered, thus making them difficult to distinguish between real and GAN generated images, while the images from StarGAN, SPADE and StyleGAN are more separable, thus resulting in higher accuracies in the leave-one-out experiment.

3.4.7 Comparison with State-of-the-art

We compare our proposed approach with various state-of-the-art methods [19–21] on the CycleGAN dataset. In [19], Marra et al. proposed the leave-one-category-out

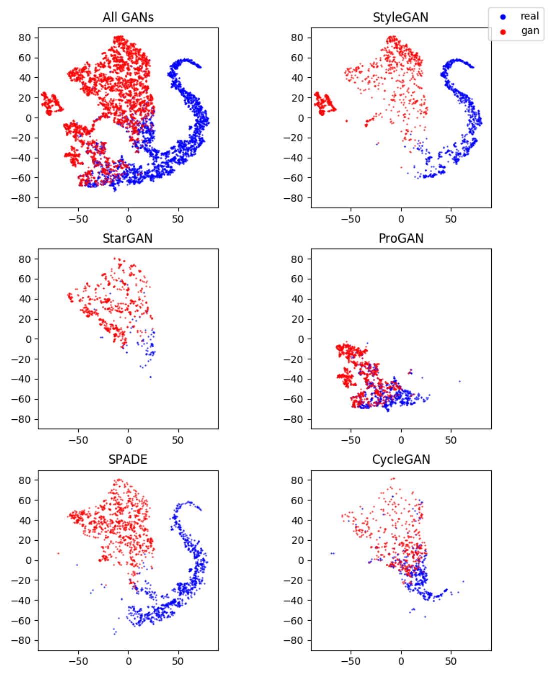


Figure 3.6: Visualization of images from different GAN datasets using t-SNE [7].

benchmark test to see how well their methods work when one category from the CycleGAN dataset is kept for testing and remaining are kept for training. The methods they evaluated are based on steganalysis, generic image manipulations, detection of computer graphics, a GAN discriminator used in the CycleGAN paper, and generic deep learning architecture pretrained on ImageNet [8], but fine tuned to the CycleGAN dataset. Among these the top preforming ones were from steganalysis [25,81] based on extracting features from high-pass residual images, a deep neural network designed to extract residual features [26] (Cozzolino2017) and XceptionNet [1] deep neural network trained on ImageNet but fine-tuned to this dataset. Apart from Marra et al. [19], we also compare our method with approaches including Nataraj et al. (Nataraj2019) [20], which uses co-occurrence matrices computed in the horizontal direction, and Zhang et al.(Zhang2019) [21], which uses spectra of up-sampling artifacts used in the GAN generating procedure to classify GAN images.

Table 3.7: Number of images per class

	Train	Val	Test
Authentic	1,612,202	42,382	42,397
StarGAN	28,062	738	711
CycleGAN	17,265	439	439
ProGAN	70,286	1833	1,881
SPADE	138,075	3,717	3,704
StyleGAN	766,045	20,220	20,158

Table 3.6 summarizes the results of our proposed approach against other state-of-the-art approaches. Our method obtained the best average accuracy of **0.9817**, when compared with other methods. Even on individual categories, our method obtained more than 0.90 on all categories.

3.4.8 Tackling newer challenges like StyleGAN2

Apart from generalization, we tested our method on 100,000 images from the recently released StyleGAN2 [90] dataset of celebrity faces. The quality of these images were much better than the previous version and appeared realistic. When we tested on this dataset without any fine-tuning, we obtained an accuracy of 0.9464. This shows that our approach is promising in adapting to newer challenges. We also fine-tuned to this dataset by adding 100,000 authentic images randomly chosen from different GAN datasets, thus our new dataset comprised of 100,000 authentic images and 100,000 StyleGAN2 images. Then, we split this data into 40% training, 10% validation and 50% testing. When we trained a new network on this dataset, we obtained a validation accuracy of 0.9984 and testing accuracy of 0.9972, thus also confirming that our approach can be made adjustable to newer GAN datasets.

3.4.9 GAN Attribution/Classification

While the primary area of interest is in determining the authenticity of an image, an immediate extension would be to determine which GAN was used. Here we perform an additional experiment on GAN class classification/attribution as a 6-class classification problem, the classes being: Real, StarGAN, CycleGAN, ProGAN, SPADE/GauGAN and StyleGAN. The number of output layers in the CNN was changed from 1 to 6, and output with the largest value was selected as the estimate. A breakdown of the number of images per class for training, validation and testing is given in Table 3.7. First, the network was trained where the input co-occurrence matrices were computed on the whole image. The training procedure was kept the same as with all other tests in the paper, with the exception of using a batch size of 60, and 10 images from each class per batch. This encouraged the network to not develop a bias towards any particular GAN for which we have more training data. First we consider the images as they are provided in the datasets. The classification results are shown in the form of confusion matrices in Table 3.8. For convenience, we also report the equal prior accuracy, equal to the average along the diagonal of the confusion matrix. This equal prior accuracy can be interpreted as the classification accuracy if each class is equally likely. We obtain an overall classification accuracy (considering equal priors) of 0.9654. High classification accuracy was obtained for most categories. StyleGAN had comparatively lower accuracy but still more than 90%, being mostly confused with SPADE/GauGAN and CycleGAN. These results show that our approach can also be used to identify which category of GAN was used.

Next, we trained the network using a patch size of 128×128 as input, and repeated the experiment. This is to see how well our method can be used for detection, localization as well as classification. The classification results are shown in Table 3.9. Now, we obtain an overall classification accuracy (considering equal priors) of 0.8477 (a drop of 12% when compared to full image accuracy). High classification accuracy was obtained for StarGAN, CycleGAN and ProGAN, while SPADE/GauGAN and StyleGAN had comparatively lower accuracies. These could be due to many factors such as the number

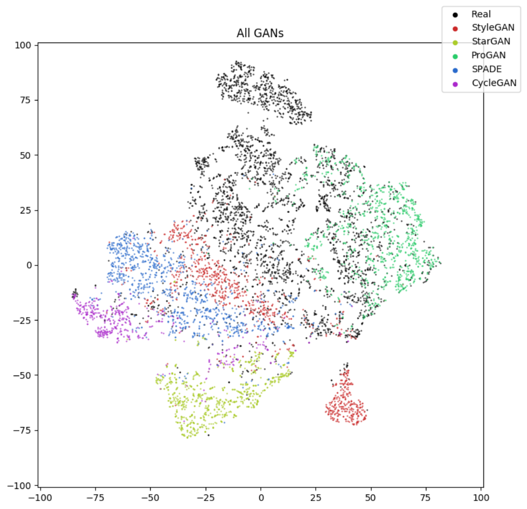


Figure 3.7: t-SNE visualization of 6 classes: Real, StyleGAN, StarGAN, ProGAN, SPADE/GauGAN and CycleGAN

Table 3.8: Confusion matrix on images from GAN datasets without any pre-processing on the full image. Equal prior accuracy of 0.9654.

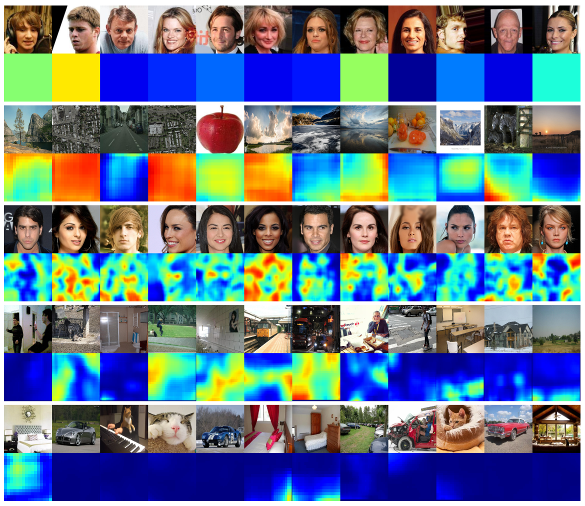
		Predicted Label					
		Real	StarGAN	CycleGAN	ProGAN	SPADE	StyleGAN
GT Label	Real	0.975	0.000	0.000	0.016	0.002	0.006
	StarGAN	0.000	0.976	0.014	0.000	0.010	0.000
	CycleGAN	0.000	0.000	0.964	0.000	0.036	0.000
	ProGAN	0.000	0.000	0.000	1.000	0.000	0.000
	SPADE	0.001	0.000	0.019	0.000	0.975	0.005
	StyleGAN	0.007	0.000	0.022	0.000	0.068	0.902

of test images per class, patch size, and the authentic image datasets that were used for training in generating these GAN images.

In Table 3.10 we repeat the same experiment (with patch size 128×128) but with images that were randomly preprocessed with JPEG quality factors of 75, 85, 90, or no JPEG compression, with each of the four preprocessing methods equally likely. For this experiment, the overall classification accuracy drops slightly to 0.8088 due to the impact of JPEG compression.

For the multi-class experiment trained without JPEG compression, we repeat the t-SNE visualization procedure. Figure 3.7 shows all data-points on a single plot. These visualizations further support the results from the classification experiment.

Figure 3.8: Localization heatmaps of (a) Real images and (b) GAN images from different GAN datasets (top to bottom): ProGAN [2], StarGAN [3], CycleGAN [4], StyleGAN [5], and SPADE/GauGAN [6].


(a) Real Images

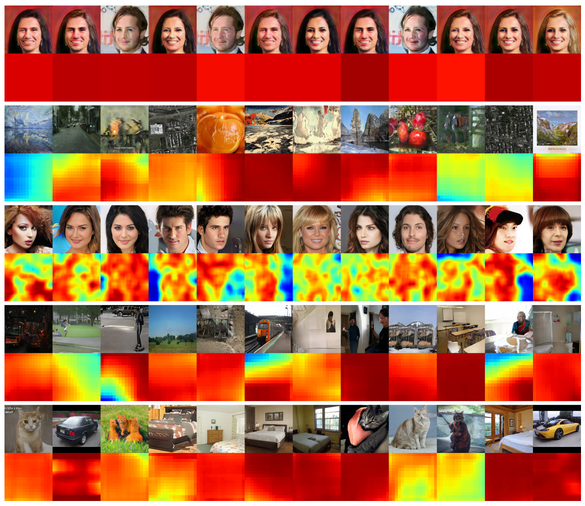

(b) GAN Images

Table 3.9: Confusion matrix on images from GAN datasets without any pre-processing on 128×128 patches. Equal prior accuracy of 0.8477.

		Predicted Label					
		Real	StarGAN	CycleGAN	ProGAN	SPADE	StyleGAN
GT Label	Real	0.826	0.003	0.016	0.021	0.066	0.068
	StarGAN	0.000	0.933	0.054	0.000	0.006	0.006
	CycleGAN	0.000	0.002	0.959	0.002	0.032	0.005
	ProGAN	0.000	0.002	0.008	0.981	0.004	0.005
	SPADE	0.001	0.025	0.210	0.008	0.728	0.029
	StyleGAN	0.003	0.025	0.101	0.009	0.203	0.659

Table 3.10: Confusion matrix with JPEG compression (128×128 patches). Equal prior accuracy of 0.8088. The images were preprocessed using a JPEG factor of 75, 85, 90, or no compression. Each of these four possible preprocessing functions was randomly selected with equal probability for every image.

		Predicted Label					
		Real	StarGAN	CycleGAN	ProGAN	SPADE	StyleGAN
GT Label	Real	0.741	0.005	0.020	0.026	0.103	0.104
	StarGAN	0.006	0.927	0.023	0.000	0.031	0.012
	CycleGAN	0.009	0.014	0.892	0.007	0.074	0.005
	ProGAN	0.002	0.003	0.009	0.973	0.007	0.007
	SPADE	0.075	0.015	0.095	0.009	0.765	0.042
	StyleGAN	0.114	0.021	0.059	0.008	0.243	0.555

3.4.10 Localization

Figure 3.8 show two example localization outputs. The image is processed in overlapping patches, with a particular stride and patch size. A co-occurrence matrix is then extracted for each patch, and passed through the CNN to produce a score. For pixels which are a part of multiple patches, the scores are simply the mean of all of the patch responses. These two examples use a patch size of 128, and a stride of 8. We can see that the heatmaps are predominantly blue for real images and predominantly red for GAN generated images. This further supports that our method can be effectively used for GAN localization.

3.5 Adversarial Attack Setup

This next section discusses several methods for adversarially attacking these detectors. All of the work in this chapter was published in [15]. Here, we only focus on the case of binary detection. However, most of these methods could be trivially generalized to the other models. The case of binary detection provides a simpler framework to test out these broadly applicable methods.

In addition to adversarially attacking the co-occurrence based method, I provide attacks for two other methods as well. The DFT method from [21], and a baseline with no feature, only a deep learning network. Figure 3.9 shows an abstract outline of these models. This section describes the 3 methods we investigate for the feature extraction step.

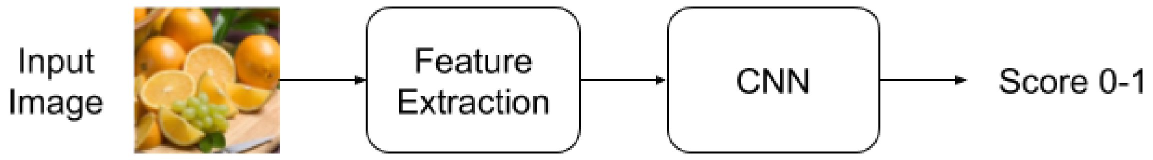


Figure 3.9: High-level diagram of the detection architectures. Sections 3.5.1, 3.5.2, and 3.5.3 describe the different feature extraction methods investigated. Section 3.8.2 describes the different CNN architectures tested.

3.5.1 Co-Occurrence

Unless stated otherwise, we will use the horizontal co-occurrence matrix defined by Nataraj *et al.*. For each channel in the RGB input image, represented by the array X, we produce a 2D histogram of horizontal pixel pairs:

$$C_{i,j} = \sum_{k,l} \delta(X_{k,l} - i) \cdot \delta(X_{k,l+1} - j) \tag{3.2}$$

Where $\delta(\cdot)$ is the Kronecker delta function:

$$\delta(n) = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{otherwise} \end{cases} , n \in \mathbb{Z} \tag{3.3}$$

Each co-occurrence matrix is then scaled into the range $[0, 1]$, re-stacked in the channel dimension, and passed to a deep learning classifier.

3.5.2 DFT

As proposed by Zhang *et al.*, we will use the DFT of the image as an input to a deep learning classifier. The processing steps for the DFT based method are as follows:

1. Get the centered, unitary DFT of the input image
2. Take the magnitude of the DFT

3. Apply the function $f(x) = \log(x + 10^{-6})$

4. Shift and scale into the range [-1,1]

All steps are consistent with those used in X. Zhang *et al.*, except for the inclusion of a small constant in step 3, to prevent a log(0) case. They use a ResNet34 architecture pretrained with ImageNet weights for detection.

3.5.3 Direct

This method will pass the image directly to a deep learning classifier, with only affine scaling as a preprocessing step. The following ImageNet means and standard are applied, as per the torchvision documentation [91]:

$$\text{mean} = 255 \cdot [0.485, 0.456, 0.406], \tag{3.4}$$

$$\text{std} = 255 \cdot [0.229, 0.224, 0.225] \tag{3.5}$$

For their direct method, Wang *et al.*used ResNet50 pretrained on ImageNet [92].

3.6 Adversarial Attack Methods

In creating an adversarial attack, there are several levels of knowledge that a potential adversary may have.

The first is white-box. This assumes total knowledge of a system by an adversary. An example of this might be openly publishing your detection code. This would allow full knowledge and experimentation with your system by an adversary.

The second is black-box. This is the weakest assumption, and is not directly addressed for this GAN detection work. However, such an attack is included later in this thesis, in the work on DeepFake detection. This setup assumes that the adversary has no knowledge of your system, but a limited ability to query it. For example, a web server where users can get a GAN detection score for 100 images a day.

The third is gray-box. This term is simply used to represent an adversary with partial knowledge about your system. For the specific usage, this would mean knowledge about the features used for GAN detection. For example, the adversary may know that the model uses a co-occurrence feature, or a DFT feature. As a real world use case, the web server from the black-box scenario may decide to advertise the high level method they are using to detect fake images. Such advertisements may increase user trust in a product, if they know what tools are being used. Or even, if there is published research to back it up. However, this will also give an adversary additional information about your system.

While using a feature like a co-occurrence matrix or DFT adds explainability to the CNN result, there is a question whether or not using some hard coded feature could introduce an additional avenue for an adversary to attack.

The next section discusses the main contribution of my work, the adversarial gray box attack on the co-occurrence based detector. Then, this is generalized to white box, and the DFT detectors.

3.7 Adversarial Gray Box attack

This section discusses the adversarial gray-box attack on the co-occurrence feature, the main contribution of [15]. The later part of this section then generalizes the attack to DFT based features, and to a PGD attack.

3.7.1 Adversarial Attack Methods

For our gray-box attack, we assume that it is known that co-occurrence matrices are the only feature used for detection, but we have no knowledge about the deep learning model used on these matrices. We also assume that the adversary has an arbitrary set of real images at their disposal. The goal of the adversary will be to modify each GAN image by some small amount, such that the co-occurrence matrix of the adversarial image is close to, if not exactly equal to, the co-occurrence matrix of a real image.

The adversarial, real, and GAN images will be represented by X_A, X_R, and X_G respectively. $F(\cdot)$ is the co-occurrence function, $\mathcal{L}_1(\cdot,\cdot)$ and $\mathcal{L}_2(\cdot,\cdot)$ are the loss functions to be defined later, and λ is a user-defined constant. The adversarial image will be proposed as:

$$X_A = \arg\min_{\tilde{X}} \mathcal{L}_1(F(\tilde{X}), F(X_R)) + \lambda \mathcal{L}_2(\tilde{X}, X_G) \tag{3.6}$$

3.7.2 Distinctions Between Co-Occurrence and Histograms

Given the formulation in equation 3.6, it is worth noting that the Earth Mover's Distance (EMD) solves a similar optimization problem, and that there exist efficient approximations [93]. For a pair of 2 dimensional histograms and for some pre-defined cost function between bins, this would produce a minimal cost transformation from one histogram to the other. However, non-edge pixels will appear twice in the co-occurrence histogram, once as the left pixel in the pair, and again as the right. Therefore, entries in the co-occurrence matrix cannot be individually manipulated to achieve this optimal transport, without inadvertently changing values at another location in the histogram.

To handle this entanglement between pairs, we use gradient descent to find an approximate minimum. This will require the functions F, $\mathcal{L}_1$, and $\mathcal{L}_2$ to be differentiable.

The original definition of the co-occurrence function was over only integer inputs, posing a problem for the differentiability requirement. The next few sections break down the details of how F, $\mathcal{L}_1$, and $\mathcal{L}_2$ are selected.

3.7.3 Differentiable Extension of Co-Occurrence Function

In creating a differentiable extension of the co-occurrence matrix, we impose the following requirements:

1. For integer inputs, $F(\cdot)$ must be equivalent to the original co-occurrence function.
2. The sum of the histogram bins should equal the number of input elements.
3. For all input elements, the contribution to each bin must be non-increasing with respect to distance from that bin.
4. It must be differentiable over $\mathbb{R}^2_{[0,255]}$.

Given the original co-occurrence formulation in equation 3.2, a simple extension would be to define a new one-dimensional function $f(\cdot)$ which will interpolate the delta function's integer values:

$$C_{i,j} = \sum_{k,l} f(X_{k,l} - i) \cdot f(X_{k,l+1} - j) \tag{3.7}$$

Equation 3.7 consists only of additions, multiplications, and $f(\cdot)$, making gradient calculation straightforward. From this equation, the previous four requirements can be simplified into these requirements on $f(\cdot)$:

1. $f(x) = \delta(x),\ x \in \mathbb{Z}$
2. $f(x) = 1 - f(x-1),\ \forall x \in \mathbb{R}_{[0,1]}$

3. $\frac{df}{dx} \leq 0$ for $x > 0$, and $\frac{df}{dx} \geq 0$ for $x < 0$

4. $\frac{df}{dx}$ should be defined for all $x \in \mathbb{R}_{[-255,255]}$

The combination of constraints 1 and 3 will require that $f(x) = 0$ for $x \notin (-1, 1)$. Therefore, each pixel pair will contribute to at most 4 bins. This fact was taken advantage of in implementation, as opposed to computing the entire summation in equation 3.7. Both the triangle and raised cosine shown below were tested as interpolation functions:

$$\text{tri}(x) = \begin{cases} 1 - |x|, \text{ if } |x| < 1 \\ 0, \text{ otherwise} \end{cases} \tag{3.8}$$

$$\text{raised\_cos}(x) = \begin{cases} \frac{1+cos(\pi x)}{2}, \text{ if } |x| < 1 \\ 0, \text{ otherwise} \end{cases} \tag{3.9}$$

For the triangle function, derivatives at $x = -1, 0, 1$ are undefined, so the average of the left and right derivatives is used. Raised cosine gave better results experimentally, and will be used for the remainder of the tests.

3.7.4 Co-Occurrence Loss Function

In this section, we provide intuitive reasoning and experimental justification for our selection of $\mathcal{L}_1$. We we first give motivating examples for the 1D and 2D histogram cases. “Source” will correspond to the GAN input, “target” to the real input, and “solution” to the adversarial solution. For all of these examples, we will assume $\lambda = 0$.

One-Dimensional Example

Consider the case with a source of $[1, 2, 3]$, a target of $[2, 3, 4]$, and $\lambda = 0$. We would now like gradient descent to push the source towards the target, making their histograms equal. In this example, all derivatives should be negative.

Shown in Figure 3.10 are plots of loss for different loss functions, varying one input at a time. Consider the loss values as x_1 is moved from 1 to 4. For L1, there is a constant loss from $x_1 = 1$ to 3. This is compared to L2, where loss fluctuates in the same region. Given that the histogram will have a constant L1 norm, and that L2 loss is less for the vector [1/2,1/2] than [0,1], the L2 loss function tended to get stuck between integer values. For this reason, we will focus on L1 loss.

Looking at the top left graph in Figure 3.10, the loss for x_1 decreases only after passing the threshold of 3. Using point-wise loss, the vacancy at 4 can only pull values which are within the $(-1, 1)$ support region of $f(\cdot)$. To alleviate this, we instead compute loss on a multiscale pyramid of the histograms, with a downsampling factor of 2 in each step.

To combine the multi-scale losses, a simple weighted sum is used. Weights are set equal to the downsampling factor at each level. These weights are selected as the cost of moving pixels between bins at each layer in the pyramid scales with respect to the downsampling factor. Results using the image pyramid loss are shown in the right column of Figure 3.10. When gradient descent is run on the different loss functions for the 1D case, the results in Figure 3.11 are produced.

Two-Dimensional Examples

A 2D example is shown in figure 3.12, which also demonstrates the necessity of random noise. Often we need points initialized to the same value to split into different outcomes, which cannot occur with deterministic gradient descent.

This two-dimensional histogram test was repeated over 100 iterations, with both source and target vectors containing 8 elements uniformally sampled from $\mathbb{Z}^2_{[0,7]}$. For the L1 pyramid with Gaussian noise, all 100 tests successfully converged from the source to

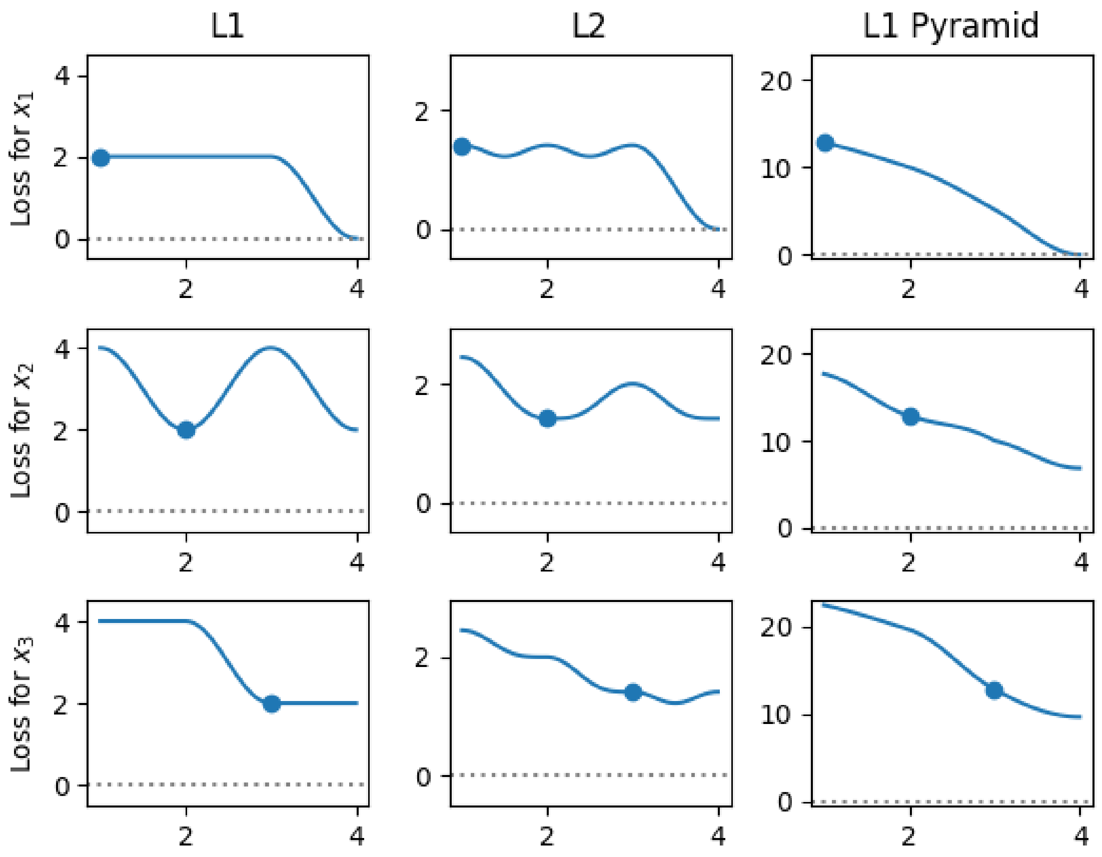


Figure 3.10: Loss functions with respect to value in each index for 1D loss functions, on source of [1,2,3] and target of [2,3,4]. Ideally, we would like the gradients for each index, at initialization, to be negative. This will push each value in the source towards the target. L1 loss would often get stuck on the plataeus, L2 would get stuck at the minima in-between integer values, and the L1 pyramid converged more easily. See figure 3.11 for convergence results.

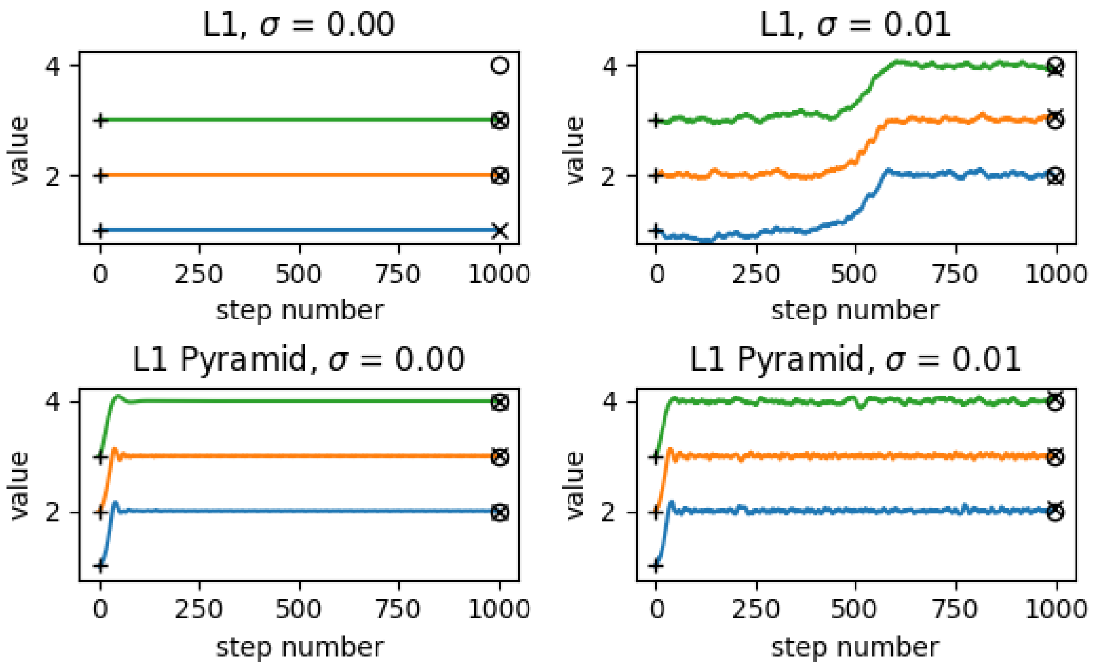


Figure 3.11: Plot of each value in the source vector with respect to step number, when solving with a source of [1,2,3] and target of [2,3,4]. Top left: without random noise, all points are stuck in flat regions, no change in values. Top right: the addition of noise allows for the algorithm using L_1 norm to gradually drift towards the target. Bottom row: Both with and without noise, the algorithm converges over 14 times faster than the top right case.

target. This is compared to only 8 for L1 with Gaussian noise.

Extension to Co-Occurrence on Images

For an 8-bit image, a 9 layer pyramid is used, with downsampling factors from 1 (None) to 256. To implement the blurring and downsampling steps in the co-occurrence image pyramid, we rely upon the original interpolation function defined for the co-occurrence. By dividing the input image by the downsampling factor before computing co-occurrence, lower resolution co-occurrence matrices can be produced. The full loss function is shown in equation 3.10.

$$\mathcal{L}_1(F(\tilde{X}), F(X_R)) = \sum_{n=0}^{8} 2^n \left\| F\left(\frac{X_A}{2^n}\right) - F\left(\frac{X_R}{2^n}\right) \right\|_1 \tag{3.10}$$

3.7.5 Image-Space Loss Function

In equation 3.6, only the $\mathcal{L}_2$ and λ terms are left to be defined. For consistency with the co-occurrence loss, L1 distance is chosen for the image-space loss. The λ parameter remains as a user selected parameter, and several values were tested experimentally.

3.7.6 Implementation Details

Ideally, we would like to choose source-target pairs with similar color values for optimization. For example, we would not want to force a GAN image with a green grassy background to have the same color distribution as a real image of a blue ocean. To do this, we divide the data into blocks of size approximately 900, and for each GAN image, select the real image whose EMD over the 1D RGB histograms is closest to that of the GAN image.

With the pairs selected, we can then run our gradient descent algorithm. The solution

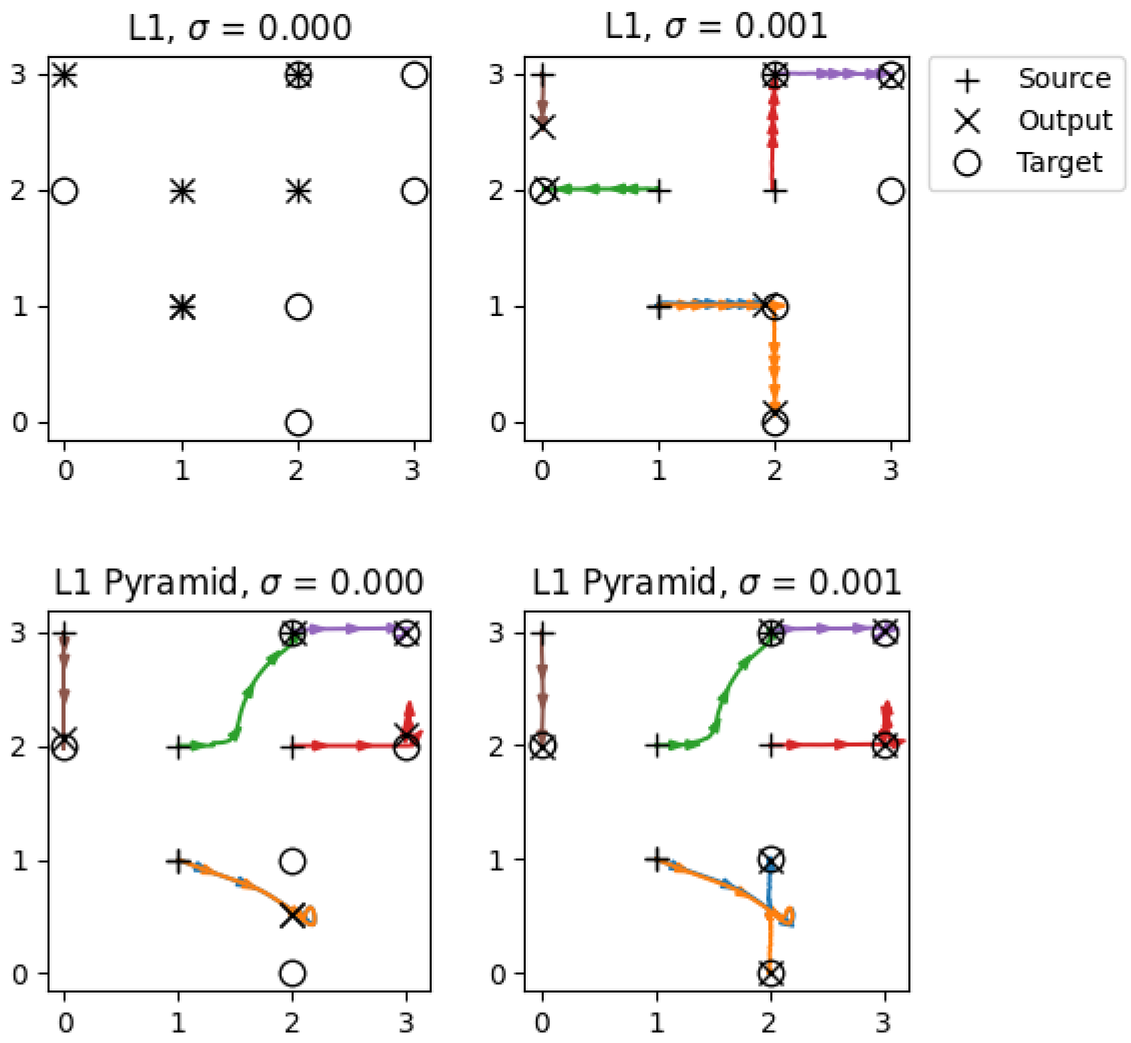


Figure 3.12: Example applying point-wise and pyramid loss to a 2D histogram gradient descent problem. Each graph is a parametric plot, with step number as the parameter, showing the path of each 2D input from source to solution. Each color represents a different 2D input element. Top left: with L_1 loss and no noise, all gradients are 0. Top right: With noise, L_1 finds a sub-optimal minima, where not all target points are reached. Bottom left: Without added noise, the pyramid loss almost converges to a global minima. However, the two source points originating from (1,1) need to split and fill different target points. With deterministic gradient descent, this splitting will not occur. Bottom right: A proper solution is found with pyramid loss and noise.

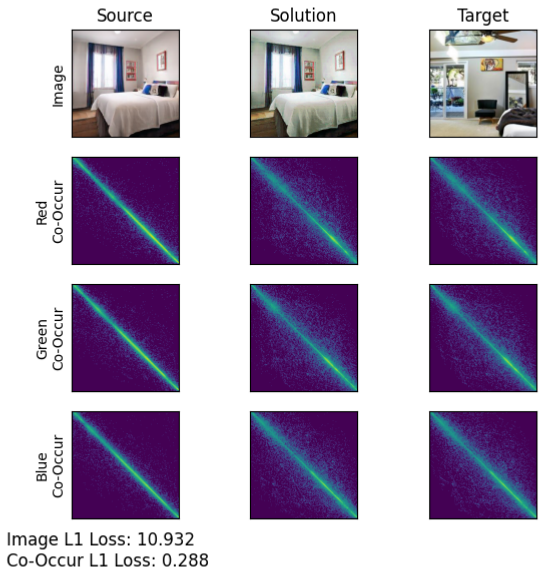


Figure 3.13: Example solution found by our algorithm, for the given real and GAN images.

is initialized with the source image. We use a standard gradient descent, with a learning rate of 0.01, and momentum of 0.9. This is done in 3 sequential epochs, with 200, 50, and 50 steps. For the first 2 epochs, Gaussian noise with standard deviation of 0.01 is added to the image. No noise is added in the last epoch. The solution is rounded after each epoch.

When run on an Nvidia 1080 Ti, the algorithm took approximately 30 seconds per 256x256 image. However, up to 3 processes could be placed on a single GPU, so 6 adversarial images could be produced every minute.

Quantitative results are shown in figure 3.15. For comparison, average L_1 loss between the co-occurrence matrices of the source target pairs was 0.90, and L_1 loss between source and target images was 52.7. For this real data, it cannot achieve a perfect match between the real and adversarial co-occurrence matrices. Two examples are given in figures 3.13 and 3.14. The effect of this slight mismatch is investigated experimentally.

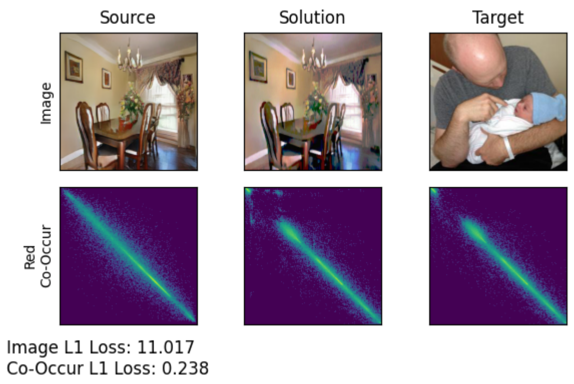


Figure 3.14: Another example solution found by our algorithm, and corresponding red channel co-occurrence matrices. In the top-left corner of the solution co-occurrence a square artifact can be seen.

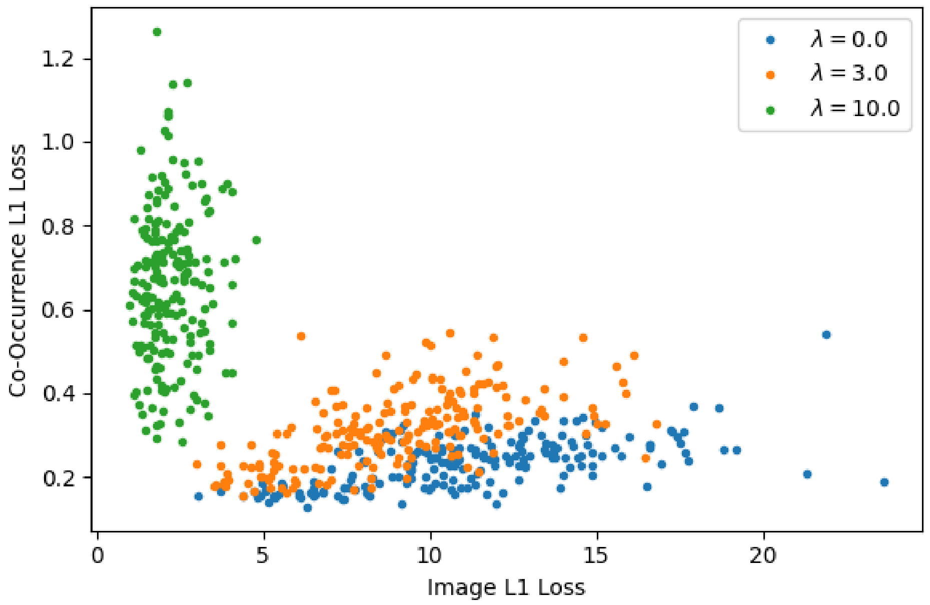


Figure 3.15: Testing of the co-occurrence algorithm on our dataset described in section 3.8.1 on 200 images for 3 different λ values. Smaller λ values will force the co-occurrence matrix of the adversarial image closer to that of the real image, at the cost of greater perturbation.

3.7.7 Gray-Box DFT

This method follows a similar formulation to the co-occurrence gray-box attack. A real image is obtained in addition to the GAN image, and the detection feature of the adversarial image is made to be similar to the real image, while minimizing the distance from the original GAN image. We rely upon the intuition that the defining features of the GAN in the DFT domain are concentrated away from the DC axes. To estimate this high-frequency noise signal, we use the same filter as Kirchner in his work on resampling [94], with a centered DFT given in equation 3.11:

$$\mathcal{F}(f) = \frac{1}{4}\begin{bmatrix} 3 & 0 & 3 \\ 0 & 0 & 0 \\ 3 & 0 & 3 \end{bmatrix} \tag{3.11}$$

To produce an adversarial image, we solve the following:

$$X_A = \arg\min_{\tilde{X}} \left\| f * \tilde{X} - f * X_R \right\|_2^2 + \lambda^2 \left\| \tilde{X} - X_G \right\|_2^2 \tag{3.12}$$

Application of the Fourier transform turns this problem into one of weighted least squares, and can be solved as:

$$\mathcal{F}(X_A) = \frac{\mathcal{F}(f)^2 \cdot \mathcal{F}(X_R) + \lambda^2 \mathcal{F}(X_G)}{\mathcal{F}(f)^2 + \lambda^2} \tag{3.13}$$

This is done between randomly selected real and GAN images, for different λ values in the experimental portion of this chapter.

3.7.8 White-Box PGD

With the co-occurrence, DFT, and direct methods all being differentiable pytorch functions, we run the L_∞ PGD algorithm on all GAN images on each method [95]. This is done using the advertorch library [96]. Default parameters are used, with a maximum distortion of 1, maximum step size of 2/40, and 40 total iterations, running on pixels in the range $[0, 255]$. Pixels are rounded after completion of PGD.

3.8 Adversarial Attack Experiments

While the previous section outlined a theoretical backing for the attack, it needs to be verified in practice. The next section details the experiments done to verify the attack's effectiveness in different settings.

3.8.1 Datasets

Our dataset consists of 4 different GANs, drawing from a variety of image datasets and tasks. Image counts are given in table 3.11. Each group is divided into a 70/15/15 train/val/test split. These groups are then further split in half; the first for training and testing of models, and the second for generating adversarial samples. All images were center-cropped to 256×256.

3.8.2 Neural Network Selection

Neural network selection was done experimentally, and results are shown in table 3.12. All models were trained with 16 real and 16 GAN images per batch, for 16 epochs. An Adam optimizer was used, with default parameters of 0.001 for the learning rate, and (0.9,0.999) for the betas [86]. The final model weights for testing are selected from the

Architecture	Dataset	Total Count
CycleGAN [4]	apple2orange [97]	3,000
	horse2zebra [97]	3,000
	summer2winter [4]	3,000
	cityscapes [98]	3,000
	cezanne [4]	3,000
	monet [4]	3,000
	ukiyoe [4]	3,000
	vangogh [4]	3,000
ProGAN [2]	CelebHQ [99]	24,000
SPADE [100]	ADE20K [101]	12,000
	COCO-Stuff [85]	12,000
StyleGAN [5]	LSUN Bedroom [102]	8,000
	LSUN Car [102]	8,000
	LSUN Cat [102]	8,000
Total		96,000

Table 3.11: Combined number of real and fake samples from each data subset. For all subsets, the number of real and fake examples is equal. In all, there were 48k real and 48k GAN images used.

Method	Co-Occurrence		DFT		Direct	
Initialization	ImNet	rand	ImNet	rand	ImNet	rand
ResNet18	0.979	0.974	0.904	0.888	0.980	0.829
ResNet50	0.979	0.977	0.864	0.900	0.976	0.866
ResNet101	0.424	0.569	0.503	0.500	0.519	0.503
ResNet152	0.500	0.668	0.495	0.500	0.500	0.499
ResNeXt50	0.978	0.975	0.882	0.907	0.986	0.853
Inception V3	0.944	0.500	0.948	0.708	0.990	0.949
MobileNet	0.978	0.974	0.949	0.919	0.996	0.989

Table 3.12: Overall accuracy of different networks on a balanced test set of real and GAN. Each row represents using a different deep learning architecture. Three detection methods are shown as the first column headers. The second shows results with either ImageNet weights or random initialization.

epoch number on which validation loss was the lowest.

The ImageNet pretrained networks did better than those with random initializations in almost all cases. As the larger networks did not provide noticeable improvements on this dataset, we will use pretrained ResNet18 and pretrained MobileNet for the remainder of tests. We chose ResNet18 to generate the PGD samples, given that ResNets were used by both Zhang *et al.*and Wang *et al.*.

	Real	GAN	GB CO $\lambda = 0.0$	PGD CO
ResNet18	0.979	0.984	0.030	0.000
MobileNet	0.976	0.981	0.039	0.083

Table 3.13: Test set accuracy of co-occurrence based detectors, without adversarial retraining. Gray-box (GB) co-occurrence (CO) samples are generated as described in section 3.7. PGD co-occurrence examples are produced using ResNet18.

3.8.3 Testing on Adversarial Samples

We then evaluated the detectors chosen in the previous section on the co-occurrence adversarial examples, with results shown in Table 3.13. For both detectors, the gray-box co-occurrence attack drops the GAN detection rate from approximately 98% to less than 4%, with no knowledge of the deep-learning model used. As expected with the PGD attack, accuracy on the exact model being attacked drops to 0. However, accuracy on MobileNet drops to only 8%; more than twice what was achieved with the gray-box attack.

3.8.4 Adversarial Training

Next we adversarially trained the same networks using different subsets of the adversarial samples. The labels remain binary, with real images in one class, and all GAN images, including adversarial GAN images, in the other.

For data balancing, we maintain an equal number of positive and negative samples. Within the positive sample class, each of the sub-types is sampled equally. For example, in the test using all gray-box co-occurrence adversarial examples, we used 16 real, 4 GAN, and 4 from each of the 3 gray-box co-occurrence classes. For the set of all adversarial images, a batch size of 40 is used so the batch can be evenly divided. For all other cases, batch size remains 32.

		Test			
		Real	GAN	CO $\lambda = 0$	CO PGD
Train	Real, GAN	0.979	0.984	0.030	0.000
	Real, GAN, All Adv	0.901	0.971	0.970	1.000

Table 3.14: Results for only the ResNet18 co-occurrence detector. The rows show results with and without adversarial retraining.

			Real	GAN	Co-Occur Gray-Box			DFT Gray-Box			PGD		
					$\lambda = 0.0$	$\lambda = 3.0$	$\lambda = 10.0$	$\lambda = 0.003$	$\lambda = 0.01$	$\lambda = 0.03$	Co-Occur	DFT	Direct
ResNet18	Co-Occur	No Adv*	0.979	0.984	0.030	0.019	0.332	0.624	0.627	0.627	0.000	0.863	0.854
		No Adv	0.976	0.980	0.032	0.027	0.358	0.906	0.883	0.852	0.000	0.907	0.878
		GBCO 0.0	0.972	0.966	0.957	0.992	0.932	0.473	0.480	0.476	0.144	0.822	0.794
		GB-CO 3.0	0.974	0.972	0.424	0.985	0.999	0.702	0.685	0.666	0.311	0.923	0.910
		GB-CO 10.0	0.985	0.973	0.121	0.826	0.997	0.360	0.402	0.444	0.000	0.890	0.876
		All GB-CO	0.964	0.951	0.947	0.999	0.997	0.415	0.448	0.472	0.104	0.883	0.858
		All GB-DFT	0.968	0.984	0.047	0.029	0.222	0.998	0.997	0.994	0.000	0.893	0.880
		All PGD	0.974	0.974	0.031	0.045	0.449	0.832	0.783	0.713	1.000	0.996	0.995
		All Adv	0.901	0.971	0.970	0.999	0.999	0.993	0.987	0.969	1.000	0.981	0.984
	DFT	No Adv*	0.874	0.934	0.824	0.796	0.860	0.311	0.277	0.239	0.853	0.284	0.888
		No Adv	0.882	0.971	0.801	0.808	0.914	0.235	0.236	0.218	0.971	0.165	0.910
		All GB-CO	0.868	0.973	0.962	0.967	0.965	0.213	0.200	0.214	0.964	0.420	0.831
		All GB-DFT	0.905	0.831	0.851	0.820	0.881	0.950	0.946	0.913	0.925	0.273	0.780
		All PGD	0.959	0.930	0.809	0.784	0.875	0.114	0.089	0.089	0.943	0.992	0.935
		All Adv	0.771	0.834	0.983	0.984	0.984	0.890	0.962	0.956	0.951	0.858	0.870
	Direct	No Adv*	0.974	0.985	0.782	0.792	0.932	0.286	0.355	0.384	0.989	0.989	0.000
		No Adv	0.947	0.989	0.868	0.862	0.965	0.279	0.369	0.427	0.992	0.991	0.013
		All GB-CO	0.971	0.966	0.995	0.995	0.994	0.290	0.428	0.524	0.976	0.975	0.032
		All GB-DFT	0.930	0.964	0.774	0.783	0.896	0.998	0.999	0.998	0.959	0.960	0.090
		All PGD	0.979	0.976	0.849	0.848	0.963	0.193	0.287	0.363	0.991	0.988	0.997
		All Adv	0.936	0.981	0.993	0.993	0.994	0.996	0.999	0.999	0.989	0.988	0.999
MobileNet	Co-Occur	No Adv*	0.976	0.981	0.039	0.027	0.456	0.576	0.556	0.553	0.083	0.945	0.914
		No Adv	0.974	0.985	0.029	0.020	0.359	0.695	0.672	0.647	0.036	0.947	0.924
		All Adv	0.952	0.974	0.955	0.999	0.999	0.996	0.994	0.986	1.000	0.996	0.995
	DFT	No Adv*	0.955	0.962	0.632	0.630	0.748	0.182	0.187	0.190	0.941	0.222	0.891
		No Adv	0.928	0.970	0.524	0.539	0.770	0.192	0.189	0.176	0.926	0.217	0.910
		All Adv	0.865	0.911	0.981	0.982	0.985	0.956	0.959	0.919	0.983	0.926	0.944
	Direct	No Adv*	0.995	0.997	0.553	0.635	0.845	0.204	0.189	0.190	0.979	0.984	0.787
		No Adv	0.993	0.996	0.640	0.694	0.808	0.088	0.043	0.030	0.849	0.861	0.744
		All Adv	0.989	0.984	0.993	0.996	0.999	0.999	1.000	0.999	0.997	0.998	0.998

Table 3.15: Comprehensive table showing test set results on all datasets, for many training combinations. Each row represents a different detector and/or training set, and each column the data set tested on. The PGD data is generated using the models labeled with an asterisk.

Full Results

All results are shown in table 3.15, with a summary of just the co-occurrence results in table 3.14. These results show test accuracy on only one group at a time. When comparing across rows, it is important to note any changes in accuracy on real images in addition to changes in GAN performance.

From this table, we summarize the results as follows:

1. Without adversarial retraining, all adversarial attacks would generally decrease performance on all detectors.

2. Adversarially training on one attack method does not generally improve performance against other methods.

3. After adversarial training, the models which are most different from the assumption in the adversarial attacks performed best. Notably, MobileNet trained on all adversarial images achieved over 98% on all subsets.

For all co-occurrence based detectors which were not trained on the gray-box co-occurrence attack, accuracy was less than 5% for $\lambda = 0$. This included models which were trained on all other adversarial attacks. This indicates that most co-occurrence detectors not trained on this particular attack would remain highly vulnerable. After retraining, accuracy on the $\lambda = 0$ class was only slightly lower than the regular GAN class. The difference was more significant in the MobileNet co-occurrence detector.

Also noteworthy is that the DFT and direct classifiers performed, on average, 18% worse on the GB-CO $\lambda = 0$ set than the original GAN images, if they were not trained against the GB-CO attack. Performance generally improved back to baseline levels after retraining.

We repeat the tests from table 3.15 using JPEG compression ($Q = 75$), and have put these results in the supplementary materials.

3.8.5 Other Tests

Reversed Gray-Box Co-Occurrence Attack

Though less useful as a real-world attack, the GB-CO method can also be used to generate real images which will be classified as GAN. With the target and source switched, we produced a test set of adversarial real images, and tested on the regular ResNet18 co-occurrence detector (row 1 in table 3.15). 95.9% of the images were misclassified as GAN.

Real	GAN	Adv C-Band
0.974	0.992	0.131

Table 3.16: Results of cross-channel co-occurrence detector on cross-channel co-occurrence method described in section 3.8.5, without adversarial retraining. Adversarial images were generated using the gray-box method, using this new co-occurrence formulation.

Cross Channel Co-Occurrence Detector

Recently a paper was posted on arXiv from M. Barni *et al.*claiming to have improved the original co-occurrence GAN detector by including cross-channel co-occurrence matrices [103]. Their cross-band co-occurrence matrices for a red-green pair are defined in equation 3.14, assuming HWC convention on X. This is repeated for the red-blue and green-blue pairs. For spatial co-occurrence, they instead use diagonal pairs, shown in equation 3.15. After producing these 6 co-occurrence matrices, they are stacked in the channel dimension, and passed to a ResNet18 classifier.

$$C_{i,j} = \sum_{k,l} \delta(X_{k,l,1} - i) \cdot \delta(X_{k,l,2} - j) \tag{3.14}$$

$$C_{i,j} = \sum_{k,l} \delta(X_{k,l} - i) \cdot \delta(X_{k+1,l+1} - j) \tag{3.15}$$

We also modified the co-occurrence gray-box attack equation in 3.7 to accept the 6 different pairs used in M. Barni *et al.*. We ran the algorithm to produce an adversarial test set for the cross-band co-occurrence detector. Results are shown in table 3.16. The gray-box attack remains effective against detectors using this other co-occurrence feature.

3.9 Conclusion

In this chapter, we presented two main projects. The first used extensive testing to benefit the advantages of using a co-occurrence matrix in GAN detection, over a CNN alone. The second portion discussed the adversarial setting. This work proposed several new adversarial attacks against co-occurrence matrices as a feature in general. It then demonstrated the effectiveness of the attack against an unassuming network, and ways to reduce susceptibility of networks to attack.

Acknowledgements: I would like to thank and acknowledge my collaborators on the paper ”Detection, Attribution and Localization of GAN Generated Images” [14] In particular, the help of my co-author Lakshmanan Nataraj. The problem of GAN image detection was introduced to me by Lakshmanan during a summer internship at Mayachitra. The work on adversarial robustness was partially supported by NSF SI2-SSI award #1664172 and NSF HDR IDEAS2 award #1934641.

Chapter 4

RPC Verification

This next chapter discusses the usage of resampling artifacts for the task of satellite image RPC metadata verification. Our paper, pusblished in IEEE Tansactions on Information Forensics & Security, was the first work investigating the problem of RPC metadata verification [104]. In this work, I created and did initial testing on the first draft of the RPC verification model. Tejaswi Nanjundaswamy performed additional testing, and is credited with the idea to use SSIM in the algorithm, with Chandrakanth Gudavalli performing additional testing on new datasets and contributing substantially to writing the final paper.

In the following we describe the problem background, then the detection method, and finally experimental results.

4.1 Background

While forensic analysis on digital images have been well studied [49, 105, 106], there are fewer studies in detecting manipulations in satellite images [107] and even lesser when it comes to the detection of tampering of metadata. Satellite images, for the most part, are orthorectified using rational polynomial coefficients (RPC) and Digital Elevation Models (DEM). The RPC coefficients define a best-fit mapping from latitude, longitude, and elevation to pixel coordinates. These coefficients are determined from the camera location, orientation, and parameters intrinsic to the imaging system. Such RPC coefficients are provided in the image metadata. The DEMs contain a dense grid of elevation measures at different points around the globe, and are freely available online. Unlike RPC coefficients, DEMs are not unique to a particular satellite image. The combination of the RPC coefficients with the DEMs allows for the transformation of the captured image into an orthorectified view.

RPC metadata associated with an image is essential in obtaining the pixel coordinates of a given object or a given lat-long in the unrectified image. Tampering could be used by adversaries to mask the true locations of objects or other geo-features of interest. Tampering of RPC metadata associated with orthorectified satellite imagery raises questions and suspicions on the authenticity of image content as well. This paper addresses the problem of verifying RPC metadata that is attached to orthorectified satellite imagery.

Our approach to detecting the tampering is based on resampling estimation. Given an image, there exists a noise associated with it. Here noise can be treated as the deviation from ideal pinhole camera model. Once an image is resampled using an affine transformation, the noise variance fluctuates periodically across the image. DFT of noise variance in the resampled image can be calculated using the method proposed in Kirchner et. al [94], which we refer it as "Residual DFT Pattern" (a sample is shown in Figure 4.1).

The RPC metadata, together with the DEMs, define a mapping from sensor pixel space to orthorectified image space. DEMs for a given location define a mapping for orthorectified image space to elevation. The combination of RPC and DEMs together create a mapping from lat-long coordinates to sensor pixel location. This mapping therefore defines an expected resampling pattern due to orthorectification. Using the RPC + DEM

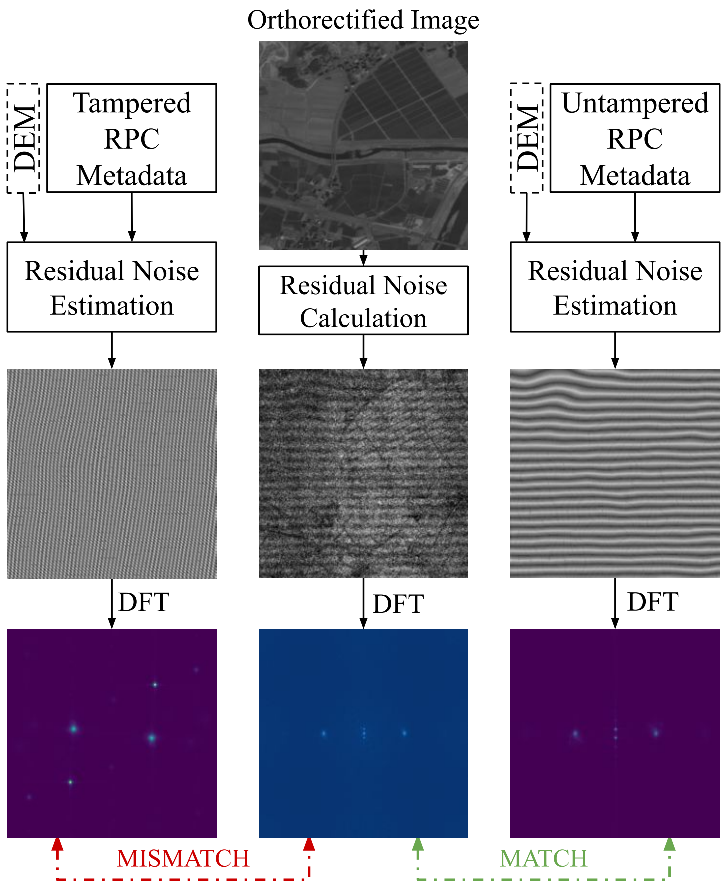


Figure 4.1: Visualization of Expected Residual DFT patterns when RPC Metadata is tampered (left) and Untampered (right), with respect to Residual DFT pattern of the image (middle).

metadata associated with the image, we resample a predefined checkerboard pattern and estimate the DFT of noise variance in the original image. We refer to this estimated DFT as "Expected Residual DFT Pattern". Any modifications to the RPC metadata will alter this expected DFT pattern. If the metadata used to resample/orthorectify the checkerboard pattern is same as the metadata used to resample the satellite image, then both the DFTs show structural similarity, as shown in Figure 4.1. Therefore, we use the structural similarity index metric (SSIM) between the two DFT patterns to verify if the associated RPC metadata is tampered on not.

Since the process of orthorectification creates a new, warped set of image pixel locations, resampling must be used to produce the orthorectified image. Altering of the RPC coefficients will affect the warping pattern used, and therefore the resampling. There are several methods used for resampling detection and/or estimation ([49, 94, 108–111]). In our proposed approach, we selected a fixed linear predictor based residual spectral analysis as described in Kirchner et. al [94]. This method offers fast prediction for large images, which makes the technique reliable for satellite images as they tend to have larger dimensions. Images that we worked on are typically of size $20,000 \times 8,000$. This method also calculates relatively unique features for a variety of scaling, rotation, and sheer factors.

4.2 Orthorectification and RPC Metadata

Orthorectification is the process of transforming an image onto its upright planimetry map by removing the perspective angle. Orthorectification is done using Rational Polynomial Coefficients (RPCs) based on empirical models that relate the geographic location (latitude/longitude, denoted by X, Y) and the surface elevation data (denoted by Z) to map the row and column positions (denoted by r, c) through two rational polynomials [112]. Satellite sensor models are empirical mathematical models that relate image coordinates (row and column position) to latitude and longitude using the terrain surface elevation. The name Rational Polynomial derives from the fact that the model is expressed as the ratio of two cubic polynomials. A single image involves two rational polynomials, one each for computing row and column position as shown in (4.1).

$$r = \frac{P_1(X,Y,Z)}{P_2(X,Y,Z)}, \quad c = \frac{P_3(X,Y,Z)}{P_4(X,Y,Z)} \tag{4.1}$$

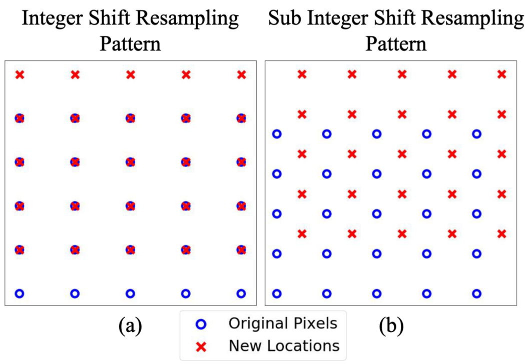

Integer Shift Resampling Pattern
Sub Integer Shift Resampling Pattern
(a)
(b)
Original Pixels
New Locations

Figure 4.2: Some example resampling patterns. Only the sub-integer shifts impact the noise variance patterns. (a) Pattern where noise variance is σ^2. (b) Pattern where noise variance is $0.25\sigma^2$.

where, P_1, P_2, P_3, and P_4 are cubic polynomials, each with 20 coefficients (which are referred as RPC Metadata) as shown in (4.2).

$$\begin{aligned} P_*(X,Y,Z) = a_{*\_1} + a_{*\_2}X + a_{*\_3}Y + a_{*\_4}Z + a_{*\_5}X^2 + \\ a_{*\_6}XY + a_{*\_7}XZ + a_{*\_8}Y^2 + a_{*\_9}YZ + \\ a_{*\_10}Z^2 + a_{*\_11}X^3 + a_{*\_12}X^2Y + a_{*\_13}X^2Z + \\ a_{*\_14}Y^3 + a_{*\_15}XY_2 + a_{*\_16}Y^2Z + a_{*\_17}XYZ + \\ a_{*\_18}XZ^2 + a_{*\_19}YZ^2 + a_{*\_20}Z^3 \end{aligned} \tag{4.2}$$

where, $*$ belongs to $1, 2, 3,$ or 4.

The coefficients of these two rational polynomials, shown in (4.1), are computed as the best fit mapping from spatial location (X, Y, Z) to pixel coordinates (r, c). This is done by considering the camera's orbital position, orientation, and corresponding physical sensor model.

Using the unrectified satellite image, its RPC Metadata, and a Digital Elevation Map (DEM) to provide the elevation values, an unrectified image is resampled to generate an orthorectified image. Figure 4.5 shows a visualization of this transformation, by passing

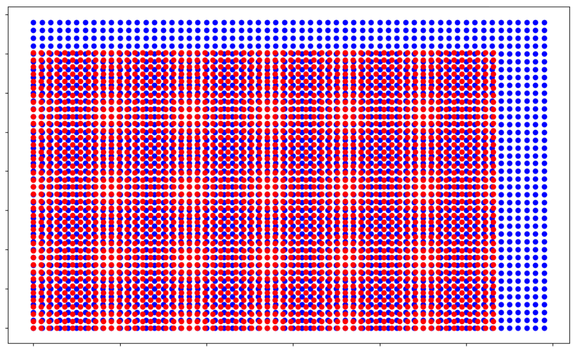


Figure 4.3: Resampling spatial mapping for a scaling example (Scale factor of 0.9; blue: original locations; red: new locations). These sampling patterns move in and out of phase with each other to form a periodic pattern. This pattern is reflected in the noise variance of the resampled image.

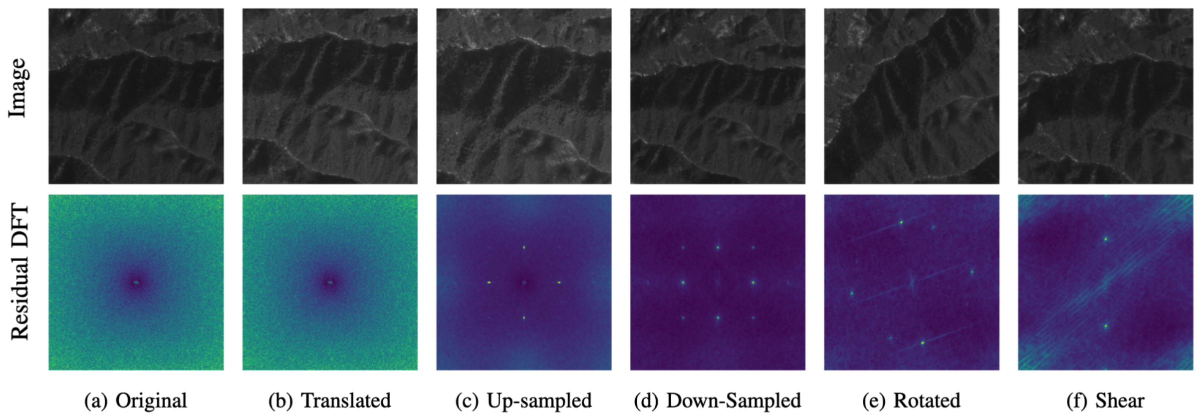


Figure 4.4: Some toy examples showing the image Residual DFT (2nd row) under different transformations.

a grid of vertical and horizontal lines through the warping function. DEMs with 30-meter resolution produced by the United States Geological Survey (USGS) are available for free download for any area in the United States, and 10-meter USGS DEMs are available in most areas of the globe.

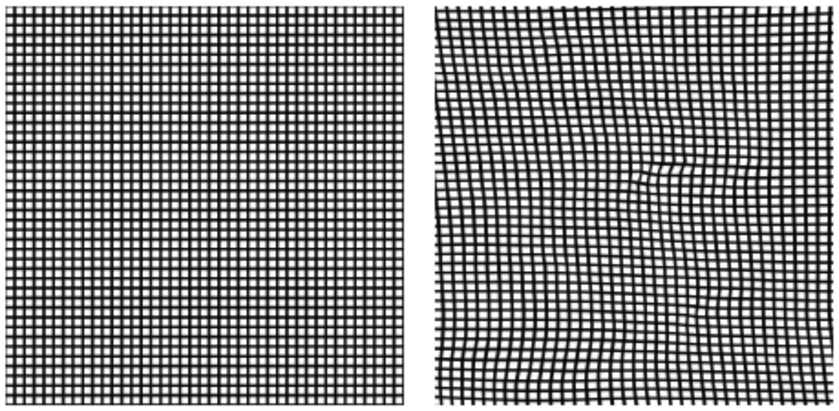


Figure 4.5: A grid before (left) orthorectification and after (right). Note the significant non-linearities introduced by the hilly region.

4.3 Method

This next section describes the methodology used for RPC metadata verification. At a high level, this involves the following steps:

- Computation of the observed resampling feature in the image
- Computation of the expected resampling feature based on the RPC metadata
- Comparison of these two features
- Setting up this method to work for large input images

Each of these points is addressed in a separate subsection below.

4.3.1 Calculation of Residual DFT Pattern

While there are several methods for resampling detection and/or estimation, base ours on a fixed linear predictor residual spectral analysis as described in Kirchner et. al [94]. This method offers faster prediction, which is essential for satellite images with sizes close to $20,000 \times 8,000$ pixels. This method first estimates an image noise signal by applying a fixed linear filter. Here noise refers to any deviation in the sample from an ideal pinhole camera model. It is shown that different resampling patterns will create unique, periodic artifacts in the noise variance, and can be analyzed through the DFT of noise variance, which we refer as Residual DFT Pattern. We now briefly describe this resampling estimation method as it is used to calculate the residual DFT pattern.

Kirchner et. al [94] assumes that the pixel noise in the image captured by the camera is zero mean and constant variance (σ^2). Resampling will cause the noise variance to fluctuate depending on the location. For the points where resampled pixels map directly onto one of the positions of original pixels, the variance will also be σ^2. Points that lie equidistant from its 4 nearest neighbors will have a noise variance of only $0.25\sigma^2$. Visual representations of both are shown in Figure 4.2. All other points in the resampling pattern will have a noise variance which lies between these two extremes.

It is shown by [94] that affine resampling methods introduce periodic patterns in the pixel noise variance present in the resampled image. An example of this is shown in Figure 4.3. This figure represents the pixel displacements that occurred due to resampling technique in which the image is scaled by a factor of 0.9. The new pixel locations will be coming in and out of phase with the original pixel locations, causing periodic patterns in the noise variance of resampled image. Given this model of the resampling process, we only need a method to estimate the pixel noise variance. Then, unique periodic patterns will be visible in the DFT of the variance estimate, which is referred as Residual DFT pattern.

To estimate the noise variance in an image, the following procedure is used by Kirchner et. al [94]. First, a high pass filter is used to remove a sufficient amount of image content. The following convolution kernel is used for the tests:

$$\begin{bmatrix} -1/4 & 1/2 & -1/4 \\ 1/2 & -1 & 1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix}$$

To estimate the noise value at each pixel, the method applies the above linear, high pass filter to remove the image content. This filtered image (denoted by e) is treated as an estimate of the noise values at each pixel. The method estimates the noise variance (denoted by p) similar to Popescu and Farid's Gaussian distribution based calculation [113] as shown in (4.3).

$$p = \lambda exp(-\frac{|e|^{\tau}}{\sigma}) \tag{4.3}$$

where $\lambda, \sigma > 0$ and $\tau \geq 1$ are controlling parameters.

The controlling parameters have been fixed by the method to $\lambda = 1, \sigma = 1$ and $\tau = 2$.

The DFT of the estimated noise variance (p) is referred to as *Residual DFT pattern*. Some toy examples showing the image residual DFT patterns under different transformations are shown in Figure 4.4.

A distinction from previous works in resampling detection is that the resampling pattern for these satellite images is not affine. While an affine transformation will produce periodic artifacts, and discrete points in the Expected DFT spectrum, the RPC+DEM resampling patterns will not. However, these patterns are very close to being locally affine for small patches, and will instead form a cloud of points in the DFT pattern.

4.3.2 Calculation of Expected Residual DFT Pattern

We calculate the DFT of "expected noise variance" at each point in the orthorectified image, which is referred to as Expected Residual DFT Pattern.

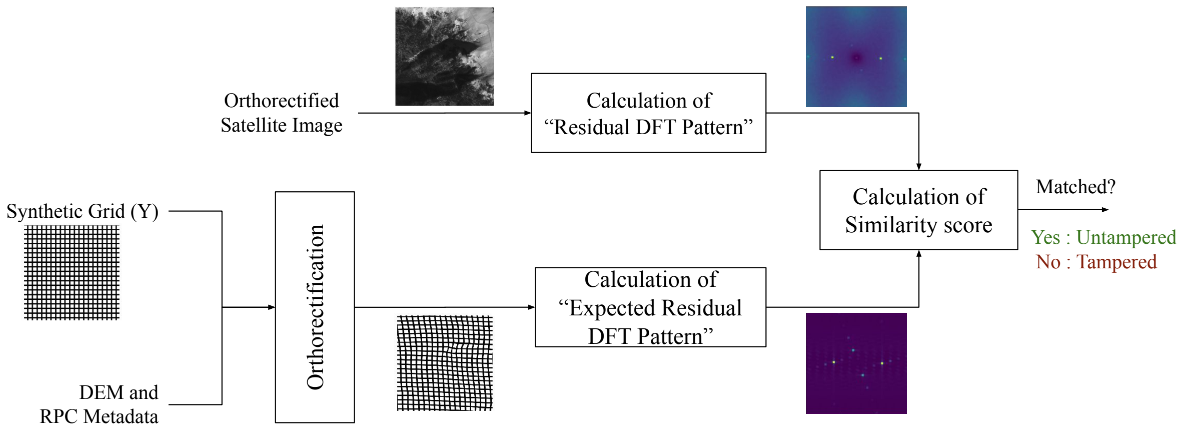

Orthorectified Satellite Image
Calculation of "Residual DFT Pattern"
Synthetic Grid (Y)
DEM and RPC Metadata
Orthorectification
Calculation of "Expected Residual DFT Pattern"
Calculation of Similarity score
Matched?
Yes : Untampered
No : Tampered

Figure 4.6: Overview of proposed framework to detect tampering in RPC Metadata.

The expected noise variance is estimated at each point in the orthorectified image by using the RPC metadata associated with the image. In order to estimate the variance at a given point, we exploit the fact that variance is inversely proportional to distance between the new orthorectified pixel location and its nearest neighbor in the unrectified image. We compute the DFT of the calculated distance, which is same as Expected DFT Pattern as the distance is inversely proportional to the variance.

We get these distances using only the forward warp function by using the following procedure. An array, Y, of the same height and width as the image (which is referred to as synthetic grid) is initialized with 4 channels. Then, it is filled with values such that any 2x2 block contains 4 orthogonal vectors as shown in Eq 4.4.

$$Y = \begin{bmatrix} a & b & a & b & \dots \\ c & d & c & d & \\ a & b & a & b & \\ c & d & c & d & \\ \vdots & & & & \ddots \end{bmatrix} \tag{4.4}$$

where, $a = [255, 0, 0, 0]$, $b = [0, 255, 0, 0]$, $c = [0, 0, 255, 0]$, $d = [0, 0, 0, 255]$.

This matrix is passed through the same transformation pipeline as the image. Assuming that the transformation can be approximated to a bilinear interpolation, from the transformed version of Y, the bilinear interpolation formula can then be reversed to calculate the distance matrix which represents the distance between each of the new orthorectified pixel location and its nearest neighbor in the unrectified image. This ensures that, it is required to provide only the forward warping function.

We calculate the DFT magnitude of this distance matrix, which we refer to as the Expected DFT pattern. This will later be compared with the residual DFT pattern to measure the structural similarity between both. Before comparing both the DFT patterns, we high pass filter the residual DFT pattern through multiplication by a cone, C, as shown in (4.5). We do it to suppress the strong and less informative low frequency components, and level out the noise floor in it.

$$C(row, col) = \sqrt[4]{(2 * row - h)^2 + (2 * col - w)^2} \tag{4.5}$$

where h, w are height and width of DFT matrix.

This entire process is summarized into a flow chart that is shown in Figure 4.6. In this section, all processes in the flow chart are defined except for computing the mismatch between two DFT patterns. Section 4.3.3 investigates different methods of computing the mismatch score.

4.3.3 Similarity score calculation between DFT Patterns

Given two DFT patterns, a metric to quantify the similarity between them is required. One of the options would be to use Mean squared error (MSE) as a metric to quantify the dissimilarity between two DFT arrays (say $\boldsymbol{x}, \boldsymbol{y}$) as shown in (4.6).

$$d_{MSE} = \frac{1}{M * N} \sum_{i=1}^{N} \sum_{j=1}^{M} {(x_{i,j} - y_{i,j})}^2 \tag{4.6}$$

where M, N are number of rows and number of columns of the DFT arrays respectively.

Otherwise, both the two-dimensional arrays can be flattened into a single dimension and compute the cosine similarity score between them as shown in (4.7).

$$cosine\_similarity = \frac{\boldsymbol{\alpha} \cdot \boldsymbol{\beta}}{\|\boldsymbol{\alpha}\| \cdot \|\boldsymbol{\beta}\|} \tag{4.7}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are the flattened DFT arrays.

Equation 4.6 and Equation 4.7 show that both MSE and cosine similarity metric take every pixel into account and perform a one-to-one comparison when computing the similarity score.

Since, we observed from our experiments that, the intensity and position of peaks in DFT patterns can vary slightly around a specific area, image similarity measures that capture "structural" similarity are of interest. We used SSIM [114] to calculate the similarity score between two DFT arrays. SSIM score between two discrete signals is calculated as follows. Let $\boldsymbol{x}$, $\boldsymbol{y}$ be the two DFT arrays that correspond to a given image patch. Let μ_x, σ_x^2 and σ_{xy} be the mean of $\boldsymbol{x}$, variance of $\boldsymbol{x}$, and the co-variance of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Approximately, μ_x and σ_x can be viewed as estimates of the luminance and contrast of x, and σ_{xy} measures the the tendency of x and y to vary together, thus an indication of structural similarity. SSIM compares luminance (l), contrast (c), and structure (s) of $\boldsymbol{x}$, $\boldsymbol{y}$ (using Equation 4.8, Equation 4.9, and Equation 4.10, respectively) and the overall similarity score is computed using Equation 4.13.

$$l(\boldsymbol{x}, \boldsymbol{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.8}$$

$$c(\boldsymbol{x},\boldsymbol{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.9}$$

$$s(\boldsymbol{x},\boldsymbol{y}) = \frac{2\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{4.10}$$

$$C_3 = C_2/2 \tag{4.11}$$

where C_1, C_2 and C_3 are small constants given by

$$C_1 = (K_1\ L)^2, C_2 = (K_2\ L)^2\ and\ C_3 = C_2/2, \tag{4.12}$$

L is the dynamic range of the pixel values (L = 255 for 8 bits/pixel gray scale images), and $K_1 << 1$ and $K_2 << 1$ are two scalar constants. The general form of SSIM between signal x and y is defined as:

$$\text{SSIM}(\boldsymbol{x},\boldsymbol{y}) = [l(\boldsymbol{x},\boldsymbol{y})^\alpha \cdot c(\boldsymbol{x},\boldsymbol{y})^\beta \cdot s(\boldsymbol{x},\boldsymbol{y})^\gamma] \tag{4.13}$$

where $\alpha,\ \beta\ and\ \gamma$ are parameters to define the relative importance of the three components. We use $\alpha, \beta, \gamma = 1$ to give equal importance to luminance (l), contrast (c), and structure (s). Hence the resulting SSIM index is given by

$$\text{SSIM}(\boldsymbol{x},\boldsymbol{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.14}$$

Since SSIM measures similarity (i.e., 1 implies matched, and 0 implies mismatched), we use $(1 - \text{SSIM}(\boldsymbol{x},\boldsymbol{y}))$ as the metric to calculate the distance (or dissimilarity) between the two DFT patterns in our experiments. Figure 4.7 and Figure 4.8 show sample scenarios explaining the reason for choosing SSIM over MSE and cosine similarity metrics respectively.

SSIM based scores were concentrated in a very small range. So we normalized the scores using a sigmoid function, $f(x)$ in Eq 4.15, to be well spread between 0 (matched) and 1 (mismatched), thereby easily differentiate matched and mismatched pairs.

$$f(x) = \frac{1}{1 + e^{-\lambda_e(x-\mu_e)}} \tag{4.15}$$

We empirically estimated λ_e, μ_e (which determine the shape of the sigmoid curve) from the outputs of experiments on larger datasets.

4.3.4 Handling of the Full Image

Given a large satellite image and associated RPC metadata, we divide the image into non-overlapping patches and compute both the DFT patterns for each patch. We calculate the SSIM score for each patch and generate a heatmap of SSIM scores for the entire image. Median of patch-wise SSIM scores is treated as the overall tampering score of the image, whereas heatmap can be used to determine where and how the match and mismatch differ.

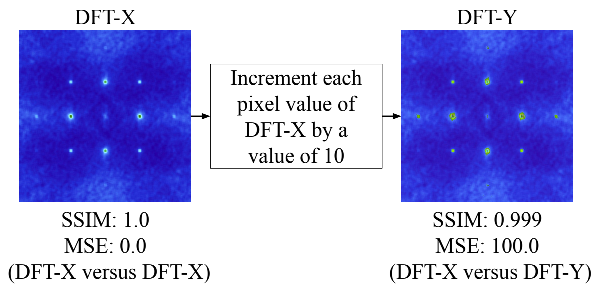


Figure 4.7: Sample demonstration to show that SSIM is more apt than MSE for our use case. Even though both the DFT patterns look alike, MSE is unable to capture the similarity unlike SSIM.

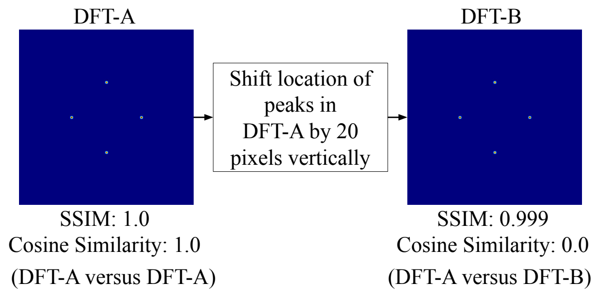


Figure 4.8: Sample demonstration to show that SSIM is more apt than Cosine Similarity for our use case. Even though both the DFT patterns look alike, cosine similarity is unable to capture the similarity unlike SSIM.

4.4 Experiments

This section describes the experiments that are carried out to verify the authenticity of satellite images using proposed technique. First is a description of the datasets used. Second, a series of experiments to verify the effectiveness of this method.

4.4.1 Dataset

Level 1B data (in GeoTIFF format, with RPC Metadata associated) from Orbview-3 satellite [115] is collected from USGS Earth Explorer [116]. For elevation maps that are required for orthorectifying the images, we used SRTM 1 Arc-Second Global data [117].

Samples from different regions of the globe with both flat and hilly terrains are used to carry out the experiments. Table 4.1 shows the number of samples collected from each region. Each sample in the dataset has pixel information and corresponding metadata associated with it. We created a dataset of tampered samples by replacing the metadata of each sample with metadata associated with one of the other samples in the dataset. We used two different approaches to create the tampered data. In the first approach we randomly exchange metadata (see Section 4.4.2). In the second approach we selected images that are overlapping spatially and captured at different time instances, and exchanged their metadata. We consider 85% overlap and 98% overlap, see Section 4.4.3

4.4.2 Random exchange of metadata

Experiments are carried out on Japan dataset by considering the collected image-metadata pairs as samples of untampered dataset. We created a dataset of tampered samples by replacing the metadata of each sample with metadata associated with one of the other samples that are randomly selected from the dataset. We calculated both DFT patterns for each sample by dividing images into non-overlapping patches of size 1024 x 1024. As described in Section 4.3.3, normalized SSIM score between DFT patterns of each patch is calculated. We set the window size parameter to 63 while computing SSIM score. Median of scores of all patches in a given image is considered as the overall tampering score of the image and this score is used as the key to detect tampering in metadata. Binary classification of these scores resulted in:

- Area under ROC curve (AUC) of 0.9969
- Accuracy of **99.15%**
- Tampered detection accuracy of 99.65% (Percentage of tampered samples that are detected as tampered)
- Untampered detection accuracy of 98.65% (Percentage of untampered samples that

Region	**Number of Images**
Japan (Hilly Terrain)	1708
Northern Europe (Estonia, Latvia, Lithuania) (Flat Terrain)	88
South America (Paraguay) (Flat Terrain)	848
West Africa (Western Sahara region) (Flat Terrain)	355

Table 4.1: Data collected from different regions of the globe to test the proposed method.

are detected as untampered)

We repeated the above experiment for various patch sizes and the corresponding results are shown in Table 4.2. As we got highest accuracy for patch size of 1024x1024, we finalized on breaking the images into 1024x1024 patches.

Patch size	**AUC**	**Accuracy (%)**
2048x2048	0.9978	98.92
1024x1024	0.9969	**99.15**
512x512	0.9951	99.00
256x256	0.9956	98.68
128x128	0.9958	98.71

Table 4.2: Performance for varying patch size

4.4.3 Spatially overlapping image metadata exchange

Experiments presented in Section 4.4.2 are carried out by randomly exchanging metadata between different samples in a dataset. But, a more purposeful manipulation is exchanging metadata between samples corresponding to the same GPS coordinates that are captured at different timestamps. To recreate this, for each image, we search for a maximum overlapping pair captured at different timestamp in the dataset. We used the image pairs with overlap percentage (overlap area/original area) above a threshold to exchange metadata between them.

With 85% overlap area threshold, only 164 images (and 328 samples) remain from the 1708 images of Japan region. Conducting tampered sample detection experiments with patch size of 1024x1024 on this small dataset resulted in:

- Area under ROC curve (AUC) of 0.9848
- Maximum accuracy of 97.85%

Region	**Overlap Area Threshold**	**Number of Samples**	**AUC-ROC**
Japan	0% (Random)	3416 (1708 images)	0.996
	85% Overlap	328 (164 images)	0.984
	98% Overlap	94 (47 images)	0.994
Northern Europe	0% (Random)	174 (87 images)	0.987
	85% Overlap	138 (69 images)	1.00
	98% Overlap	90 (45 images)	1.00
South America	0% (Random)	1696 (848 images)	0.999
	85% Overlap	518 (259 images)	0.989
	98% Overlap	246 (123 images)	0.989
West Africa	0% (Random)	710 (355 images)	0.999
	85% Overlap	134 (67 images)	0.997
	98% Overlap	52 (26 images)	1.00

Table 4.3: Experimental results on the datasets from different parts of globe. Each image stems two samples - one in the tampered dataset and the other in untampered dataset.

- Tampered detection accuracy of 98.16%
- Untampered detection accuracy of 97.55%

With 98% overlap area threshold, only 47 images (and 94 samples) remain from the big dataset of 1708 images. Classification results on this dataset resulted in:

- Area under ROC curve (AUC) of 0.9941
- Accuracy of 96.81%
- Tampered detection accuracy of 100%
- Untampered detection accuracy of 93.62%

For dataset from Japan region, results for purposeful temporal metadata exchange are slightly worse than random metadata exchange. But, this is not the case with the datasets from other regions.

4.4.4 Testing on flat regions

Orbview-3 satellite data is collected from other regions of the globe with flat regions ($<$ 500 feet variation). Tampering detection experiments are conducted for Northern Europe, South America and West Africa datasets and corresponding results are described in Table 4.3

The comprehensive results for three different flat regions of the world compared to the Japan dataset, suggest that the performance of our proposed algorithm does not suffer even when there are no significant terrain features that can add more features to the DFT pattern.

4.5 Conclusions

In this chapter, we presented a work focusing solely on the task of satellite image RPC metadata verification. An accuracy of over 99% was achieved for the satellite image as a whole. The original paper for this work was the first to investigate the problem of RPC verification. [104] Furthermore, this was one of very few works to tackle a problem in the domain of non-linear image resampling estimation or detection.

Acknowledgements: I would like to thank and acknowledge my collaborators on the paper "Resampling Estimation based RPC Metadata Verification in Satellite Imagery" [104]. In particular, the help of my co-authors Chandrakanth Gudavalli and Tejaswi Nanjundaswamy. This work was done in collaboration with researchers at Mayachitra, and supported by the National Geospatial Intelligence Agency (NGA) under Contract No. HM047619C0056.

Chapter 5

RPC Splicing

Given the success of our RPC tampering detection method, there was motivation to generalize the RPC resampling estimation method to detecting other manipulations. In this case, we look at splicing in satellite images. The next sections first present an overview of the specific problem we are trying to solve, details on how a dataset was created, and finally, experimental justification for the proposed method.

5.1 Motivation

The previous chapter demonstrated a convincing usage of resampling artifacts for the task of RPC metadata verification. The task of RPC metadata verification was specific to satellite images only, and the first work to investigate this problem. However, there also exists numerous manipulations which can be done to image pixels. These manipulations are not unique to satellite images, and countless previous works have investigated this problem for detecting these in handheld camera imagery. These methods are constrained to only have knowledge of the image pixels, no metadata.

Naive application of these tools can be applied to satellite images. With some retraining, comparable accuracy can be seen on satellite image datasets to that of the original paper. But, satellite images come with the additional information of the RPC metadata. This section demonstrates that simple signal processing methods which utilize the RCP metadata can significantly outperform the state-of-the-art deep learning methods for image splicing detection.

5.1.1 Problem Outline

As an end result, the method should take in an image + metadata package, and localize where the manipulation took place. Figure 5.1 gives a visualization of the desired output for a given input. For most of the work, it is assumed that the RPC metadata is authentic. Tests showing its effectiveness without authentic metadata are at the end of the chapter.

The presence of metadata also resolves some ambiguity with regards to which region should be marked as manipulated. For example, if an image is spliced, with half of the pixels from image A, and the other half from image B, is the region taken from image A or B the manipulated one? Here, the definition of a spliced region will be any image pixels which were not included in the original image corresponding to the metadata.

5.2 Dataset

The data collection consisted of two primary steps. First was collection of a large number of real satellite images. Second, creating a dataset of manipulated images from these which would mimic the real-world use case.

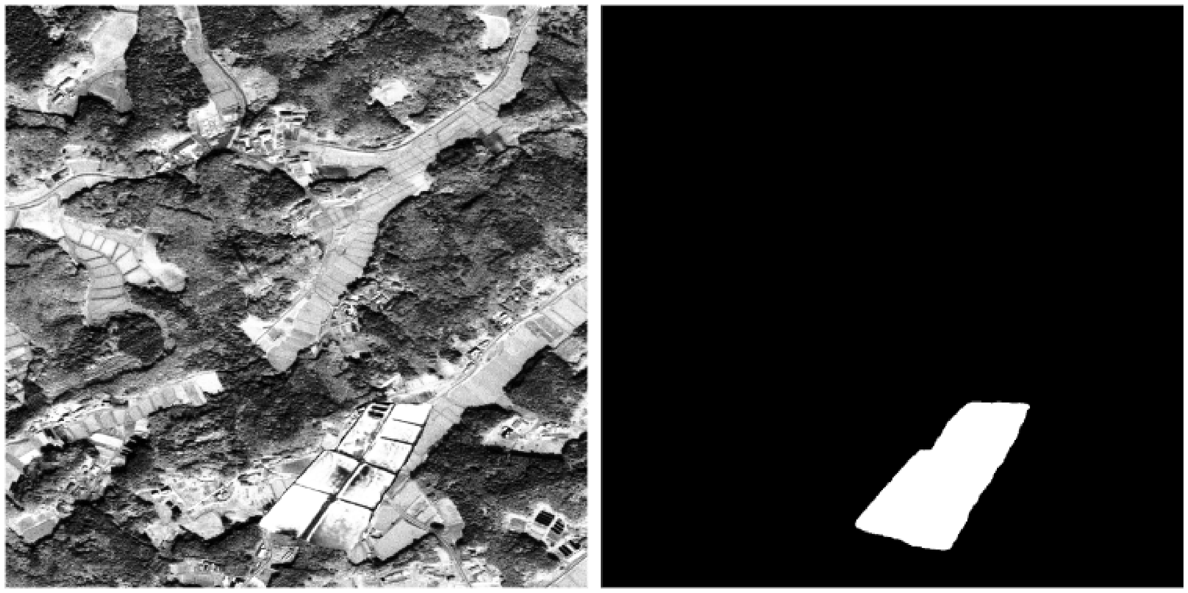


Figure 5.1: A sample manipulated image and corresponding manipulation mask.

5.2.1 Source Images

All data for this work was taken from the publicly available OrbView-3 satellite. Only panchromatic images were used, with a pixel size of approximately $1m^2$. Images were taken from 3 distinct geographic regions:

- Japan
- South America
- West Africa

Separating by region ensures that there is no crossover between the training and test sets, and provides a diversity of geography. In all parts of the process, the regions are kept separate. Splicing only happens within a region, and testing is done in a leave-one-out fashion. For example, test results for Japan will be produced by first training a model on South America and West Africa.

The data was further parsed into 2048 $\times$ 2048 patches. The native data size of 20k $\times$ 8k (160 megapixels) was far too large for the segmentation software. Furthermore, inference time on such large images using any deep learning method would be extremely long. While feasible in practice, it did not allow for fast experimentation. The choice of a 4 megapixel patch size allowed for sufficiently complex content in each patch. Generalization to the full image size is trivial, as all methods used to follow are fully convolutional.

5.2.2 Spliced Data Creation

The process consists of three major steps.

- Segmentation of objects of interest.
- Random splicing
- Automated and human quality control on manipulated images

Segmentation

Segmentation is done in the Computer Vision Annotation Tool (CVAT) [118]. Most real-world splicing manipulations would fall into one of two categories - either adding or removing information. For example, a building could be added to an image, or, greenery could be used to cover up an existing building. Good objects for both of these tasks were identified by a human annotator.

Two methods were used for segmentation of these regions of interest. The primary method was Facebook Research's Segment Anything module [119]. Segment Anything is a deep learning model trained to segment objects based on a set of point annotations. If the segmentation result is deemed to be poor, more point annotations can be added, and a better segmentation can be computed.

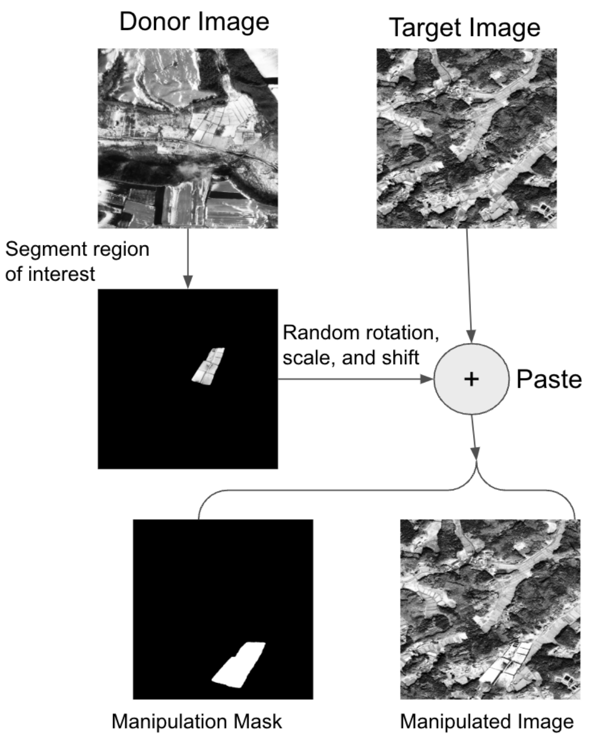


Figure 5.2: The process of creating spliced images. Two images from the same geographic region are used. One provides the background, and the other, an object to be spliced in. Random scaling, rotation, and translation is applied.

In approximately 1/5 cases, the segment anything module did not give precise enough results. In this case, manual segmentation with a polygon was used.

Splicing

A visual overview of the splicing process is shown in Figure 5.2. The selection of source and target image pairs is done randomly. 70% of the donor patches were randomly rotated. Independently, 70% were also randomly scaled, with a scale factor between 0.5 and 2. Random translation was used in all samples to place the object in a random location inn each target image.

Quality Control

After producing numerous sample images, several quality control metrics were put in place to ensure the images looked realistic. First, the following automated criteria were enforced:

- The pixel standard deviation in the target region should be at least 10. This ensures that the background is not so flat that it carries little visual information.
- The mean of the donor matches the target in the same area within +50% or -33%. In other words, the image being spliced in should be close enough in average intensity to the original.
- The standard deviation is between the donor and target matches within a multiplicative factor of 3. This threshold is more forgiving, allowing for more varied source to target transfers. For example, a building being pasted into a plain background may have relatively different standard deviations. However, vastly different values beyond 3 will stand out as obviously manipulated.

After all of the above checks, human verification was done. Given the large quantity of images, this was a 5 second check per image to sort out obvious fakes. For example, placing a building in the middle of the ocean would be filtered out in this step.

5.2.3 Final Manipulated Image Statistics

For each of the 2048 $\times$ 2048 patches, the average number of manipulated pixels was 1.75%, while the median was 0.839%. A diverse range of manipulation sizes were considered, as seen in Figure 5.3.

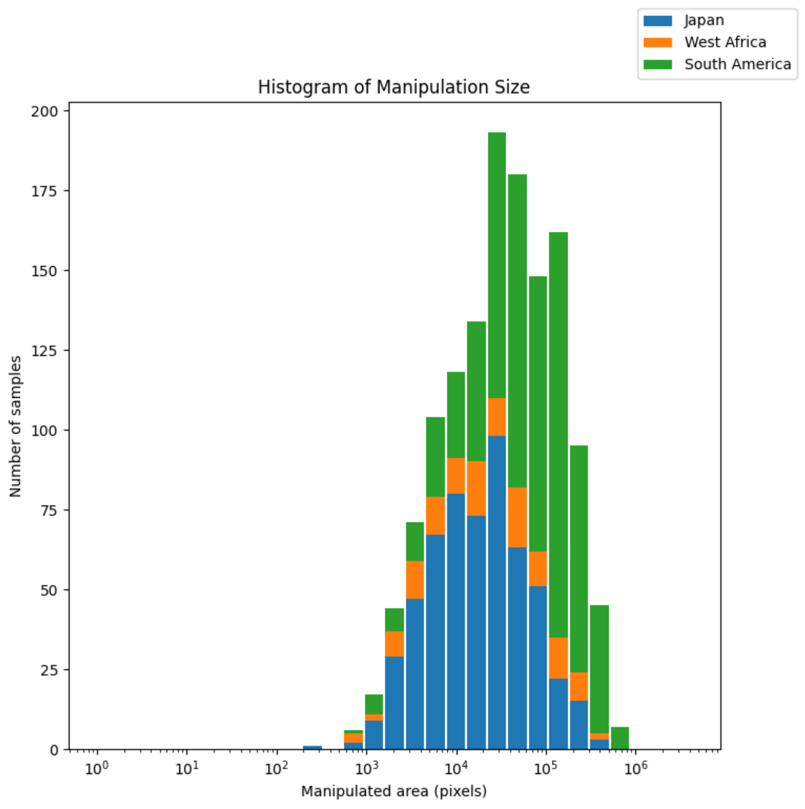


Figure 5.3: A histogram of manipulation area in pixels for each region. Note the horizontal axis is a log scale.

Country	Number of Images
Japan	560
West Africa	131
South America	634

Table 5.1: Total number of spliced images per country.

5.3 State-of-the-Art Baseline

As a SoTA baseline method, MantraNet [11] was chosen. The model was trained on 385 different image manipulations, and is designed for the task of localization. Furthermore, it is fully convolutional, and can accept the large input sizes seen in satellite imagery.

The three main contributions of this work were:

- The usage of image residual features as an input to the convolutional neural network. While this has been done before [10], this work combined many of the filters used in SoTA forensics models.
- The anomaly detection network. This layer incorporated techniques from a Z-score test to identify anomalous regions of the image.
- The optimization of both of the above layers through extensive experimentation.

Given the extensive experimentation done with the network, the next question was how well this would perform on the satellite images directly. Initial application of their pre-trained model to the satellite splicing dataset gave poor results. The AUC-ROC was slightly above random chance on all 3 test sets.

This is somewhat expected, as there are a few key factors which differentiate satellite images from those seen in normal handheld camera images. These are:

- Higher bit depth. The satellite images were 11 bit instead of the normal 8 bit.
- Lack of gamma correction.
- Lack of JPEG compression artifacts.

Test Location	Japan	South America	West Africa
AUC-ROC	0.8383	0.8351	0.8550

Table 5.2: MantraNet Test Results after fine-tuning.

Modifying the test images by applying gamma correction did improve results, but not to more than 60% AUC. However, this process is lossy, throwing away information from the smaller bits.

Significantly better results were obtained by fine-tuning the network on the new data. The strategy essentially matched that of the original MantraNet paper for fine tuning. Each training batch was half real images, and half manipulated, with a patch size of 256 $\times$ 256. For the manipulated samples, only a small portion of the 2048 $\times$ 2048 area is actually manipulated. During training, it is enforced that the 256 $\times$ 256 sub-patch must contain a portion of the manipulated region.

5.4 RPC based Pixel Alteration Localization (RPC-PAL) Model

This is a two part model, which builds upon both the MantraNet model and the RPC tampering detection model. At a high level, the RPC information can provide a coarse estimate as to where the tampering took place. The MantraNet portion provides more fine-grained segmentation.

5.4.1 From whole image to localization

The input image and checkerboard start as large 20k $\times$ 8k images, which are then cropped down into 2048 $\times$ 2048 regions for training and evaluation. To achieve localization within these crops, further patches are extracted. These are done with a square patch size of 128 pixels, and a stride of 64. Different patch sizes and strides were experimented with in the final end-to-end system. A patch size around 128 gave the best results. Smaller strides gave slightly better accuracy, at the expense of computational cost. There seemed to be little benefit to using anything smaller than 1/2 the patch size, or 64 pixels. Once these patches were extracted, and evaluated individually for similarity, they were reformed into an array, and scaled back to the original resolution using bilinear interpolation.

5.4.2 RPC + DEM Pathway

In this computational sequence, the method determines what the expected resampling pattern is in each small patch region. Once the checkerboard pattern is passed though the resampling function, using the RPC and DEM, it is parsed into patches as described above. The resampled checkerboard is used to compute the L1 distance between each pixel in the new image, and its nearest neighbor in the original image space. This is used as a proxy for the estimate of image pixel noise. Then the magnitude of the Fourier spectrum is computed as the expected local resampling signature.

5.4.3 Image Pathway

As with the RPC + DEM pathway, this one starts with the patching process. Each of these steps is run on individual patches, until the comparison step.

The steps are as follows:

- Apply the high pass filter
- Non-linear variance estimator function
- Fourier Transform magnitude

- Multiply with a cone-like high pass filter. In particular, the function $y(\omega) = \sqrt{|\omega|}$
- Divide the resulting array by the standard deviation of its values, to give the whole signal a variance of 1.

$$\begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & -1 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix} \tag{5.1}$$

5.4.4 Experimentation with Variance Estimator

Under the assumption that the noise is a zero mean, Gaussian random variable, the minimum mean squared error estimate of the variance would be equal to the square of the residual. However, there is a limit to how strongly the assumption that linear predictor residue is a good estimate of the noise. There will often be outliers, where portions of the image will have significantly larger high frequency components.

Previous works relied on a Gaussian function as a non-linearity to estimate the variance. For the simplicity in comparison, it should be noted that shifting or scaling the activation function vertically will not have any affect on the final output. The constant shift will appear only in the F(0,0) component of the Fourier Transform, which will be thrown out by the cone filter. Any scaling will be thrown out by the final normalization step of the resulting filter In comparing the square function and Gaussian, there is little difference for small deltas. For larger deltas, the Gaussian essentially clips the values at 1, doing so in a smooth way.

The followup question would be whether or not the Gaussian activation function is optimal. For large delta values, the square function grows like $O(x^2)$, where as the Gaussian grows as $O(1)$. The intuitive extension used in this work is to try a function between these two extremes. The desired function should act like $O(x^2)$ for small deltas, and $O(x^k)$ for large deltas. The k value should be between 0 and 2, the two extremes already investigated. Experimentally, values close to 1 tended to perform well, so k will be fixed at 1.

The second question is how quickly the change from $O(x^2)$ to $O(x^1)$ should happen. This can be achieved by simple horizontal scaling of the activation function. This transition point can be represented as L.

In designing a function for this task, it can be more easily be done in the derivative space. For small deltas, the derivative should be a linear slope. For large deltas, it should be a constant -1 or +1. In machine learning, this function is often referred to as "hard tanh" However, smoothness would be desired here, so a normal hyperbolic tangent will be suitable. With this as the derivative, the final function would be the integral of this:

$$f(x) = L * ln(cosh(\frac{x}{L})) \tag{5.2}$$

Experimentally, L was chosen to be 30 based on the satellite image results.

5.4.5 Matching Metric

Both the image and RPC pathways result in a grid of spectral patterns. As with the previous chapter, the SSIM metric was used. However, some additional post-processing was done to optimize performance.

While it is true that the patterns may visually look similar, the scales of the two patterns may be significantly different. For example, multiplying all pixels in the input image by a factor of 10 will increase the magnitude of the image resampling signatures by a factor of 10. But the same increase will not be seen in the RPC resampling signature.

Notably, the SSIM penalizes for this difference in scale. The eliminate this, both of the signatures are divided by their respective standard deviations, to give each signal a

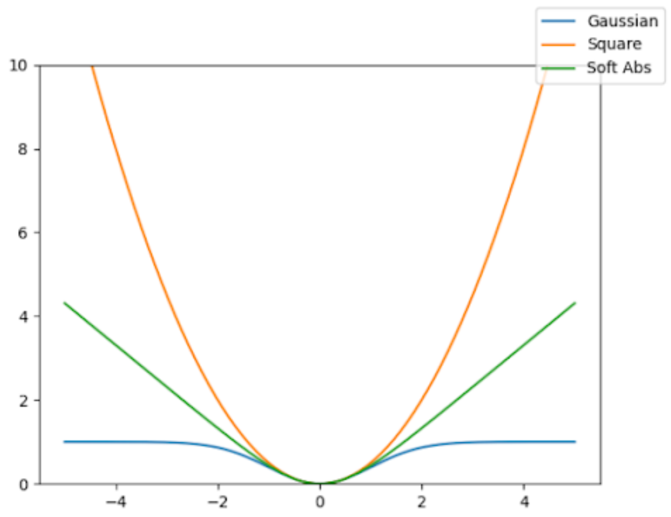


Figure 5.4: Visualization of the different non-linearities. The soft absolute value function strikes a balance between the square function which is ideal under perfect noise estimation, and the Gaussian kernel, currently used by the previous resampling works.

variance of 1. Note that this variance will be different than the one used in the SSIM calculation. In SSIM, the variance is computed over a window. In these experiments, the SSIM window size was set at 63.

5.4.6 Full Model

To produce the most accurate model, information from both the MantraNet output and RPC matching were combined. Anecdotally, the MantraNet model was capable of giving very fine-grained annotations. It had knowledge of textures and edges, and could create good object segmentation boundaries for manipulated regions. However, it suffered from a significant number of false positives, and noise outside of the manipulated region.

In contrast, the RPC based method produced accurate, but coarse masks. The intuition is that combining these would take the best of each of these. False positives from the MantraNet model could be suppressed by the RPC method, acting almost as an

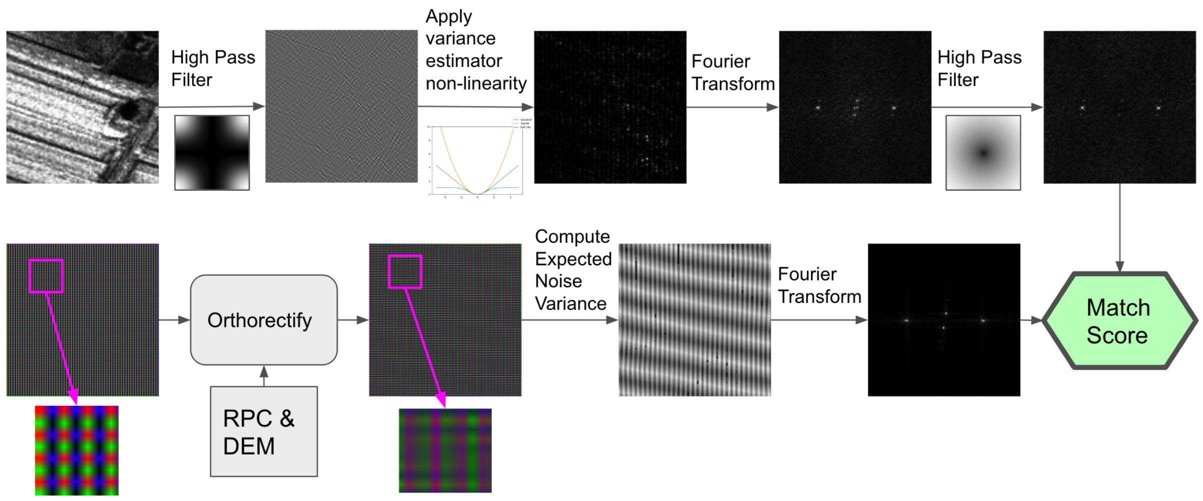


Figure 5.5: Full RPC-PAL network diagram. The details of each computational step are described in the text.

attention map for likely manipulated regions.

To combine these two models, several methods were tried. The first were pixel-wise methods. These would make a decision given only the RPC model and MantraNet outputs at a single pixel as input. Both a DNN and simple logistic regression were tried, with very similar results. The second type of method tried used whole images as input. A basic UNet was tried, but results were still less than 0.1% better than logistic regression. In the end, logistic regression was chosen. Based on the logistic regression weights, after accounting for difference in variance of the two signals, the weight on the RPC model output was around 10 times higher than the MantraNet output. Essentially, the RPC model has a much more significant impact on the final output than MantraNet. This is confirmed by the individual results of each model in the next section.

	Japan	South America	West Africa
MantraNet	0.8383	0.8351	0.8550
RPC only - Gaussian	0.7915	0.8102	0.8554
RPC only - soft abs	0.9600	0.9805	0.9677
RPC-PAL (Combined Model)	**0.9723**	**0.9813**	**0.9773**

Table 5.3: Comparison of all networks. Scores are reported as AUC-ROC. The combine RPC-PAL model out-performs all other models on each of the test sets.

5.5 Experimental Results

5.5.1 Image Pixel Manipulation Only

The testing setup was done in a leave-one-out fashion, based on locations. Results reported on each test location are from models trained on the remaining two.

Most notable in these results was the effect of the specific non-linearity used. Switching from the Gaussian to soft-abs significantly improved AUC. The combination of the RPC model and MantraNet gave modest improvements over the RPC method alone.

5.5.2 RPC tampering and Splicing

As a follow-up, an obvious attack here would be to manipulate the image RPC metadata in addition to the pixel values. This introduces a bit of a philosophical question - if the RPC data is from one image, and pixel values from two other images, which of the three pieces of data would be considered ”authentic” in the new composite image, and which should be labeled as fake. The answer more-or-less comes down to an arbitrary choice of verbiage.

The problem would be more accurately posed as finding if each image pixel is consistent with the provided RPC metadata. In the upper left quadrant of Figure 5.9 is the case of no manipulation. Both the RPC method and MantraNet are classifying almost all pixels as unlikely to be tampered.

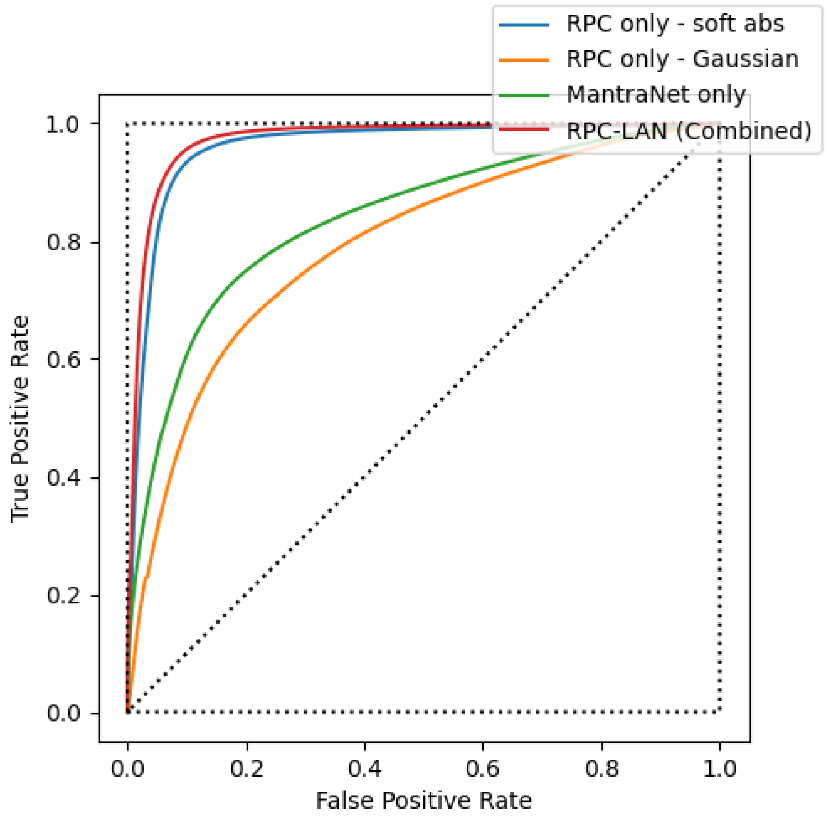


Figure 5.6: ROC curves for the Japan test set.

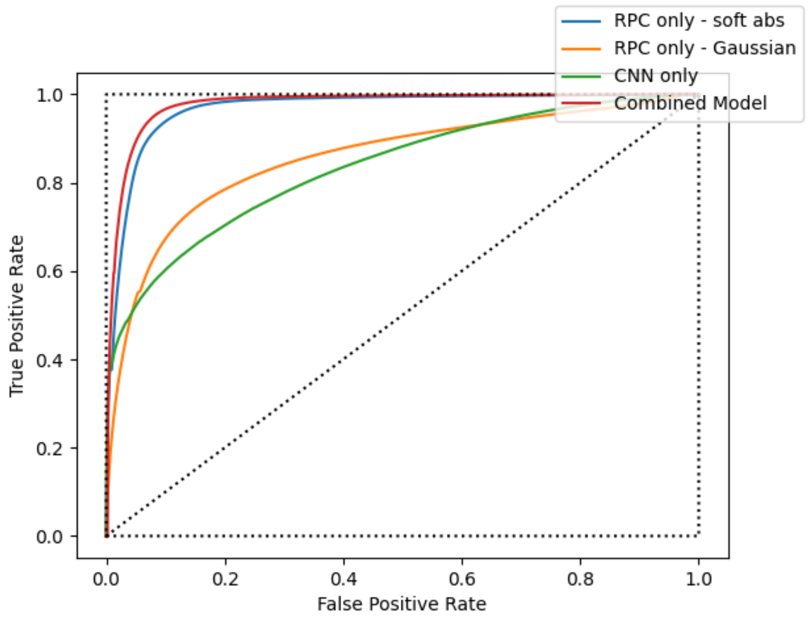


Figure 5.7: ROC curves for the South America test set.

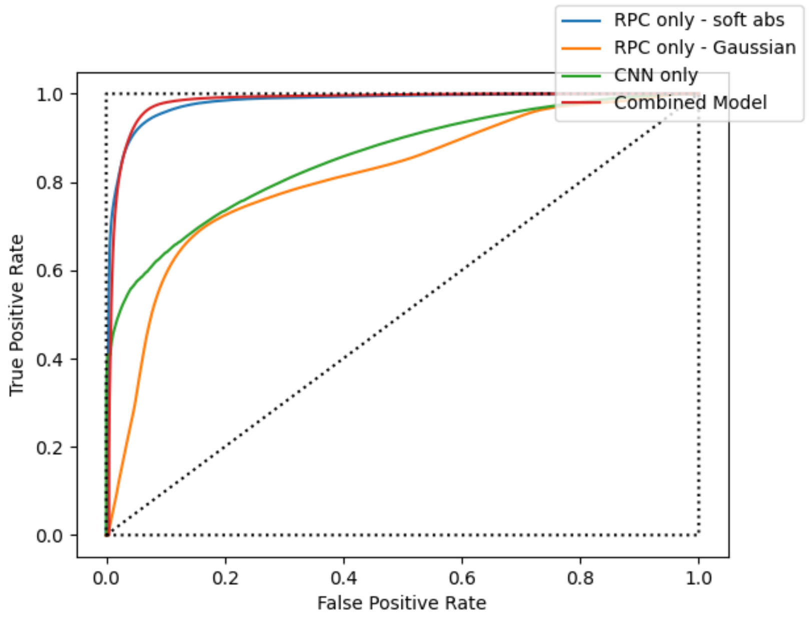


Figure 5.8: ROC curves for the West Africa test set.

In the lower left quadrant is testing on the spliced images, again with the untampered RPC. This was the case which was investigated in the previous section, and the main focus of this work. The dotted lines represent the true positives for image tampering. There is more significant separation between the histograms for the RPC only model compared to MantraNet.

Of more interest is the right hand column. At the top, is RPC manipulation only. RPC data was randomly swapped between images in the test set. There is of course no change in the MantraNet scores, as it does not use the RPC metadata. The RPC based model instead classifies almost all pixels as being tampered. This is expected and desirable, in the sense that none of the pixel noise patterns should match the expected patterns based on the RPC.

Finally, the bottom right considers simultaneous manipulation of the RPC metadata and the image pixels. Again, from left to right, the MantraNet model is agnostic to the RPC metadata, and there is no change. The RPC based model, instead, flags all pixels as being manipulated. This is consistent with what should be expected based on its derivation. As none of the pixels will be consistent with the RPC metadata, all are labeled as fake.

5.6 RPC Splicing Detection Summary

Given the previous results, it is clear that the simpler RPC based method can outperform a neural network with access to only the image pixel values. In addition to explainable, the RPC method also offers far greater efficiency. However, for full performance, both of the two can be used together in the full RPC-PAL model.

Given this, there are several ways in which this model can be utilized in real-world use cases. The RPC model can function as an initial filter, to determine if full neural network

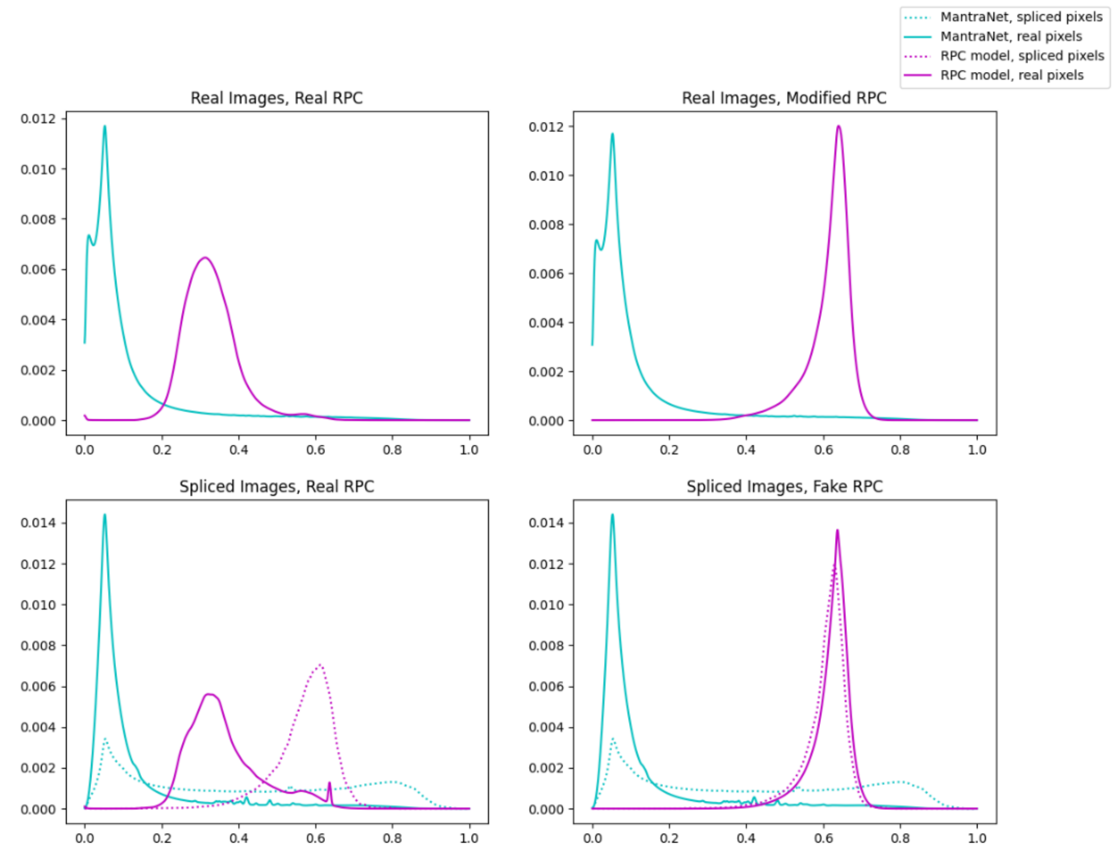


Figure 5.9: Histogram of pixel detector responses for several scenarios. The RPC based model will flag most pixels which are spliced and/or have manipulated RPC metadata.

evaluation is necessary. Depending on the size of the region identified, the MantraNet model could either be run on the whole image, or over a rectangular region which was identified as suspicious by the RPC model.

The work presented here showed the value of incorporating RPC metadata into the single task of splicing localization. However, there is reason to believe that this method could be generalized to other manipulations. For example, inpainting. By utilizing a simple signal processing feature, explainability can be assigned to model decisions. However, there likely remains a small performance gap which can be filled in by a neural network, as was done in creating RPC-PAL.

While this section only investigates satellite images, the Discussion and Conclusion Chapter offers some ideas as to how methods described here could be applied to other imaging domains.

Chapter 6

Discussions and Conclusion

This dissertation presented several works in the area of image forensics. The first of these investigated the detection of GAN generated images in Chapter 3. Then, delving into the complex issues of adversarial robustness in the context of GAN image detection. Novel adversarial attacks were presented against the co-occurrence image texture feature.

Chapters 4 and 5 discussed two image forensics applications in satellite imaging. Chapter 4 addressed the more niche problem of RPC metadata verification. Such tools are necessary to authenticate overhead imagery, as the metadata carries important information about the date, time, and location of an image. Chapter 5 further leveraged this metadata into image splicing detection. It demonstrated that the RPC patterns provided significant information about the expected resampling artifacts which should be seen in an image, and that these artifacts can be localized. When applied to splicing detection, it was demonstrated that the simple signal processing approach using RPC metadata could significantly out-perform a state-of-the-art image manipulation localization deep learning model.

6.1 DeepFake Detection

In this section, I provide a brief overview of my work in Deepfake (DF) video detection. I was the second author on the paper "Generalizable Deepfake Detection with Phase-Based Motion Analysis", with Ekta Prashnani as the lead author [120]. In a similar idea to the GAN detection work, this paper relied on a theoretically motivated feature, computed on the input, and passed to a deep learning framework.

6.1.1 Summary of the Work

Existing methods that rely on temporal information across video frames for DF detection have many advantages over the methods that only utilize the per-frame features. However, these temporal DF detection methods still show limited cross-dataset generalization and robustness to common distortions due to factors such as error-prone motion estimation, inaccurate landmark tracking, or the susceptibility of the pixel intensity-based features to adversarial distortions and the cross-dataset domain shifts. Our key insight to overcome these issues is to leverage the *temporal phase variations* in the band-pass frequency components of a face region across video frames. This not only enables a robust estimate of the temporal dynamics in the facial regions, but is also less prone to cross-dataset variations. Furthermore, we show that the band-pass filters used to compute the local per-frame phase form an effective defense against the perturbations commonly seen in gradient-based adversarial attacks. Overall, with PhaseForensics, we show improved distortion and adversarial robustness, and state-of-the-art cross-dataset generalization, with 92.4% video-level AUC on the challenging CelebDFv2 (a recent state of-the-art method, FTCN, [121] compares at 86.9%).

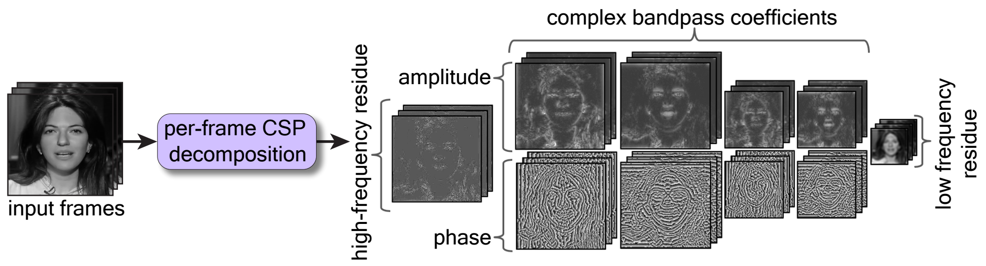


Figure 6.1: Visualization of the Complex Steerable Pyramid. The phase of the pyramid can be used to represent motion at different scales.

6.1.2 Method

The key motivation behind this method is that the facial motion in deepfake videos would be in some ways different than real ones. Real human motion is complex, and difficult to replicate. Our intuition is that our model could more easily detect abnormal motion, than a deepfake method could replicate normal motion.

The feature we used here to represent motion is the Complex Steerable Pyramid (CSP). A visualization of this is shown in Figure 6.1. The "complex" and "steerable" part of the pyramid means that motion is decomposed into separate x and y components, and any motion vector not along the x or y axis can be fully represented as a combination of x and y. The pyramid portion alludes to the fact that motion is represented at different scales. As a simple example, coarse motion might correspond to motion of the whole head, or the jaw. Finer motions could be lip or eye movements.

By restricting a neural network to only use the motion information, we argue that the network is less prone to overfitting on details particular to a dataset. This should lead to better generalization, which we demonstrate experimentally in Table 6.1.

METHOD	DFDC	VFHQ	CDFv2
Xception [122]	70.9	70.1	73.7
Multi-Attention [123]	63.0	55.0	68.0
PatchForensics [124]	65.6	-	69.6
Face X-ray [125]	65.5	-	79.5
CNN GRU [126]	68.9	66.0	69.8
Two-branch [127]	-	-	76.7
DSP FWA [128]	67.3	69.0	69.5
LipForensics [129]	73.5	90.2	82.4
FTCN [121]	74.0	84.8	86.9
PhaseForensics	**76.9**	**92.3**	**92.4**

Table 6.1: PhaseForensics results for cross-dataset generalization, where our method achieves state of the art results.

6.2 Potential Future Directions

The opportunities in image and video forensics are constantly changing. While not investigated in this work, these are several potential new directions to investigate.

6.2.1 Manipulation Detection Using Lens Correction Artifacts

In the conventional imaging process described in chapter 2, it is mentioned that there may be distortion due to the lens. This is especially apparent in the constrained environment of mobile phone cameras. A requirement for a smaller camera module leads to compromises elsewhere.

However, recent models have begun implementing distortion correction into the phone. The research question here is if the fact that the images are resampled according to a distortion patterns can be leveraged for manipulation localization.

This is a very similar setup to the splicing detection problem using RPC metadata. There is some resampling pattern which is applied to all regions of an image. Spliced objects placed into that image will not match the resampling pattern. So, successful detection of the resampling pattern can be used to identify manipulated regions.

This problem presents several unique difficulties:

- The software is closed-source, and can always change. This is the biggest barrier in researching this problem. However, sufficient examples could be collected to model the system. Deep learning would be useful here to quickly adapt to changing software.
- The resampling pattern is not known by default. However, particular patterns could be assumed. The pattern should not change too quickly, and be close to radially symmetric.
- Usage of lossy compression. JPEG compression has been studied in forensics before, and the limitations imposed by it are well known.

6.2.2 Deep Learning Generated Image Detection

While some GAN networks could be trained on a single GPU in less than a day, the newest generative models require orders of magnitude more computational power. Reports on training costs for most models cite figures in excess of $1 million USD. With this vast investment in training, there are also more restrictions on how openly available these models are.

This presents a vastly different landscape than many of the GAN detection models were developed in. The concentration of generative models into the hands of relatively few presents unique challenges and opportunities. As a challenge, it is clear that the results are becoming more visually realistic.

An opportunity is the ability to train these models such that they are easily identifiable by a watermark detection algorithm. Instead of attempting to identify small abnormalities in the image as a side effect, why not purposefully embed them ourselves.

As a first step, simply applying an off the shelf image watermarking technique would suffice for a web server. The user has no control over the source code being run, and for an everyday person, would be entirely unaware of the digital watermarking being used at all.

However, a more sophisticated user may obtain weights and code, then run the model themselves. Here, the user could simply remove any lines of code adding a watermark. Instead the watermark could be produced by the network itself. This would require, during training of the model, that the output of a watermark detector also be maximized.

In this scenario, retraining of a model would be required to remove the watermark from outputs. An open question would then be, how much fine tuning is required to remove the watermark from the model weights? Or alternatively, how effective various adversarial attacks could be at removing the watermark.

6.3 Final Thoughts

Visual media forensics remains an important area of research, as the explosion in visual media sharing continues. The growth comes from both sides, with new camera technologies, and new methods for generating fake media. Even in the 5 years of my PhD, there has been significant changes in both of these areas. These changes make media forensics an ever evolving field, full of new challenges to be explored.

Bibliography

[1] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, *arXiv preprint* (2017) 1610–02357.

[2] T. Karras, T. Aila, S. Laine, and J. Lehtinen, *Progressive growing of gans for improved quality, stability, and variation*, *arXiv preprint arXiv:1710.10196* (2017).

[3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, *Stargan: Unified generative adversarial networks for multi-domain image-to-image translation*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 2018.

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, in *IEEE International Conference on Computer Vision*, 2017.

[5] T. Karras, S. Laine, and T. Aila, *A style-based generator architecture for generative adversarial networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

[6] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, *Semantic image synthesis with spatially-adaptive normalization*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2337–2346, 2019.

[7] L. v. d. Maaten and G. Hinton, *Visualizing data using t-sne*, *Journal of machine learning research* **9** (2008), no. Nov 2579–2605.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*, in *CVPR09*, 2009.

[9] J. Lukas, J. Fridrich, and M. Goljan, *Digital camera identification from sensor pattern noise*, *IEEE Transactions on Information Forensics and Security* **1** (2006), no. 2 205–214.

[10] B. Bayar and M. C. Stamm, *A deep learning approach to universal image manipulation detection using a new convolutional layer*, in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pp. 5–10, 2016.

[11] Y. Wu, W. AbdAlmageed, and P. Natarajan, *Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9543–9552, 2019.

[12] H. Farid, *Exposing digital forgeries from jpeg ghosts*, *IEEE transactions on information forensics and security* **4** (2009), no. 1 154–160.

[13] National Geospatial-Intelligence Agency (NGA), *World geodetic system 1984*, 2014.

[14] M. Goebel, L. Nataraj, T. Nanjundaswamy, T. M. Mohammed, S. Chandrasekaran, and B. Manjunath, *Detection, attribution and localization of gan generated images*, *Electronic Imaging* **33** (2021) 1–11.

[15] M. Goebel and B. Manjunath, *Adversarial attacks on co-occurrence features for gan detection*, *arXiv preprint arXiv:2009.07456* (2020).

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[17] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, *arXiv preprint arXiv:1409.1556* (2014).

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[19] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva, *Detection of gan-generated fake images over social networks*, in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 384–389, IEEE, 2018.

[20] L. Nataraj, T. M. Mohammed, B. Manjunath, S. Chandrasekaran, A. Flenner, J. H. Bappy, and A. K. Roy-Chowdhury, *Detecting gan generated fake images using co-occurrence matrices*, in *Media Watermarking, Security, and Forensics*, IS&T International Symposium on Electronic Imaging, 2019.

[21] X. Zhang, S. Karaman, and S.-F. Chang, *Detecting and simulating artifacts in gan fake images*, *arXiv preprint arXiv:1907.06515* (2019).

[22] K. Sullivan, U. Madhow, S. Chandrasekaran, and B. S. Manjunath, *Steganalysis of spread spectrum data hiding exploiting cover memory*, in *Security, Steganography, and Watermarking of Multimedia Contents VII*, vol. 5681, pp. 38–47, International Society for Optics and Photonics, 2005.

[23] K. Sullivan, U. Madhow, S. Chandrasekaran, and B. Manjunath, *Steganalysis for markov cover data with applications to images*, *IEEE Transactions on Information Forensics and Security* **1** (2006), no. 2 275–287.

[24] T. Pevnỳ, P. Bas, and J. Fridrich, *Steganalysis by subtractive pixel adjacency matrix*, *information Forensics and Security, IEEE Transactions on* **5** (2010), no. 2 215–224.

[25] J. Fridrich and J. Kodovsky, *Rich models for steganalysis of digital images*, *IEEE Transactions on Information Forensics and Security* **7** (2012), no. 3 868–882.

[26] D. Cozzolino, G. Poggi, and L. Verdoliva, *Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection*, in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, pp. 159–164, ACM, 2017.

[27] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, *arXiv preprint arXiv:1411.1784* (2014).

[28] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, *arXiv preprint arXiv:1511.06434* (2015).

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, *Improved techniques for training gans*, in *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[30] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, 2017.

[31] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein gan*, *arXiv preprint arXiv:1701.07875* (2017).

[32] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, *Improved training of wasserstein gans*, in *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

[33] Z. Yi, H. Zhang, P. Tan, and M. Gong, *Dualgan: Unsupervised dual learning for image-to-image translation*, in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2868–2876, IEEE, 2017.

[34] Y. Taigman, A. Polyak, and L. Wolf, *Unsupervised cross-domain image generation*, *arXiv preprint arXiv:1611.02200* (2016).

[35] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, *Learning to discover cross-domain relations with generative adversarial networks*, in *International Conference on Machine Learning*, pp. 1857–1865, 2017.

[36] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et. al.*, *Photo-realistic single image super-resolution using a generative adversarial network*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, 2017.

[37] Y. Li, S. Liu, J. Yang, and M.-H. Yang, *Generative face completion*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3911–3919, 2017.

[38] S. Iizuka, E. Simo-Serra, and H. Ishikawa, *Globally and locally consistent image completion*, *ACM Transactions on Graphics (TOG)* **36** (2017), no. 4 107.

[39] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, *High-resolution image synthesis and semantic manipulation with conditional gans*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8798–8807, 2018.

[40] M.-Y. Liu and O. Tuzel, *Coupled generative adversarial networks*, in *Advances in neural information processing systems*, pp. 469–477, 2016.

[41] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, *Invertible conditional gans for image editing*, *arXiv preprint arXiv:1611.06355* (2016).

[42] H. Farid, *Image forgery detection*, *IEEE Signal processing magazine* **26** (2009), no. 2 16–25.

[43] B. Mahdian and S. Saic, *A bibliography on blind methods for identifying image forgery*, *Signal Processing: Image Communication* **25** (2010), no. 6 389–399.

[44] G. K. Birajdar and V. H. Mankar, *Digital image forgery detection using passive techniques: A survey*, *Digital Investigation* **10** (2013), no. 3 226–245.

[45] X. Lin, J.-H. Li, S.-L. Wang, F. Cheng, X.-S. Huang, *et. al.*, *Recent advances in passive digital image security forensics: A brief review*, *Engineering* (2018).

[46] S. Walia and K. Kumar, *Digital image forgery detection: a systematic scrutiny*, *Australian Journal of Forensic Sciences* (2018) 1–39.

[47] B. Bayar and M. C. Stamm, *Design principles of convolutional neural networks for multimedia forensics*, in *The 2017 IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics*, IS&T Electronic Imaging, 2017.

[48] Y. Rao and J. Ni, *A deep learning approach to detection of splicing and copy-move forgeries in images*, in *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pp. 1–6, IEEE, 2016.

[49] J. Bunk, J. H. Bappy, T. M. Mohammed, L. Nataraj, A. Flenner, B. Manjunath, S. Chandrasekaran, A. K. Roy-Chowdhury, and L. Peterson, *Detection and localization of image forgeries using resampling features and deep learning*, in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pp. 1881–1889, IEEE, 2017.

[50] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. Manjunath, *Exploiting spatial structure for localizing manipulated image regions*, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[51] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, *Learning rich features for image manipulation detection*, *arXiv preprint arXiv:1805.04953* (2018).

[52] R. Valle, U. CNMAT, W. Cai, and A. Doshi, *Tequilagan: How to easily identify gan samples*, *arXiv preprint arXiv:1807.04919* (2018).

[53] H. Li, B. Li, S. Tan, and J. Huang, *Detection of deep network generated images using disparities in color components*, *arXiv preprint arXiv:1808.07276* (2018).

[54] S. McCloskey and M. Albright, *Detecting gan-generated imagery using color cues*, *arXiv preprint arXiv:1812.08247* (2018).

[55] H. Li, H. Chen, B. Li, and S. Tan, *Can forensic detectors identify gan generated images?*, in *APSIPA Annual Summit and Conference 2018*, 2018.

[56] A. Jain, R. Singh, and M. Vatsa, *On detecting gans and retouching based synthetic alterations*, in *9th International Conference on Biometrics: Theory, Applications and Systems*, 2018.

[57] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo, *Detecting both machine and human created fake face images in the wild*, in *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, pp. 81–87, ACM, 2018.

[58] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, *Do gans leave artificial fingerprints?*, *arXiv preprint arXiv:1812.11842* (2018).

[59] H. Mo, B. Chen, and W. Luo, *Fake faces identification via convolutional neural network*, in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, pp. 43–47, ACM, 2018.

[60] N.-T. Do, I.-S. Na, and S.-H. Kim, *Forensics face detection from gans using convolutional neural network*, in *ISITC'2018*, 2018.

[61] N. Yu, L. Davis, and M. Fritz, *Attributing fake images to gans: Analyzing fingerprints in generated images*, *arXiv preprint arXiv:1811.08180* (2018).

[62] C.-C. Hsu, C.-Y. Lee, and Y.-X. Zhuang, *Learning to detect fake face images in the wild*, in *2018 International Symposium on Computer, Consumer and Control (IS3C)*, pp. 388–391, IEEE, 2018.

[63] R. Wang, L. Ma, F. Juefei-Xu, X. Xie, J. Wang, and Y. Liu, *Fakespotter: A simple baseline for spotting ai-synthesized fake faces*, *arXiv preprint arXiv:1909.06122* (2019).

[64] F. Marra, C. Saltori, G. Boato, and L. Verdoliva, *Incremental learning for the detection and classification of gan-generated images*, *arXiv preprint arXiv:1910.01568* (2019).

[65] M. Albright, S. McCloskey, and A. Honeywell, *Source generator attribution via inversion*, *arXiv preprint arXiv:1905.02259* (2019).

[66] Y.-X. Zhuang and C.-C. Hsu, *Detecting generated image based on a coupled network with two-step pairwise learning*, in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3212–3216, IEEE, 2019.

[67] P. He, H. Li, and H. Wang, *Detection of fake images via the ensemble of deep representations from multi color spaces*, in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2299–2303, IEEE, 2019.

[68] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, and H. Proença, *Real or fake? spoofing state-of-the-art face synthesis detection systems*, *arXiv preprint arXiv:1911.05351* (2019).

[69] K. Zhang, Y. Liang, J. Zhang, Z. Wang, and X. Li, *No one can escape: A general approach to detect tampered and generated image*, *IEEE Access* **7** (2019) 129494–129503.

[70] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, *Cnn-generated images are surprisingly easy to spot... for now*, *arXiv preprint arXiv:1912.11035* (2019).

[71] J. Kim, S.-A. Hong, and H. Kim, *A stylegan image detection model based on convolutional neural network*, *Journal of Korea Multimedia Society* **22** (2019), no. 12 1447–1456.

[72] A. Jain, P. Majumdar, R. Singh, and M. Vatsa, *Detecting gans and retouching based digital alterations via dad-hcnn*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 672–673, 2020.

[73] N. Bonettini, P. Bestagini, S. Milani, and S. Tubaro, *On the use of benford's law to detect gan-generated images*, *arXiv preprint arXiv:2004.07682* (2020).

[74] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz, *Leveraging frequency analysis for deep fake image recognition*, *arXiv preprint arXiv:2003.08685* (2020).

[75] Z. Guo, G. Yang, J. Chen, and X. Sun, *Fake face detection via adaptive residuals extraction network*, *arXiv preprint arXiv:2005.04945* (2020).

[76] Z. Chen and H. Yang, *Manipulated face detector: Joint spatial and frequency domain attention network*, *arXiv preprint arXiv:2005.02958* (2020).

[77] A. E. Dirik, S. Bayram, H. T. Sencar, and N. Memon, *New features to identify computer generated images*, in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 4, pp. IV–433, IEEE, 2007.

[78] R. Wu, X. Li, and B. Yang, *Identifying computer generated graphics via histogram features*, in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 1933–1936, IEEE, 2011.

[79] B. Mader, M. S. Banks, and H. Farid, *Identifying computer-generated portraits: The importance of training and incentives*, *Perception* **46** (2017), no. 9 1062–1076.

[80] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, *Distinguishing computer graphics from natural images using convolution neural networks*, in *Information Forensics and Security (WIFS), 2017 IEEE Workshop on*, pp. 1–6, IEEE, 2017.

[81] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, *Image forgery detection through residual-based local descriptors and block-matching*, in *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 5297–5301, IEEE, 2014.

[82] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, *Inception-v4, inception-resnet and the impact of residual connections on learning*, in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[83] Z. Liu, P. Luo, X. Wang, and X. Tang, *Deep learning face attributes in the wild*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

[84] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, *Scene parsing through ade20k dataset*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

[85] H. Caesar, J. Uijlings, and V. Ferrari, *Coco-stuff: Thing and stuff classes in context*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1209–1218, 2018.

[86] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

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

[88] K. He, X. Zhang, S. Ren, and J. Sun, *Identity mappings in deep residual networks*, in *European conference on computer vision*, pp. 630–645, Springer, 2016.

[89] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[90] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, *Analyzing and improving the image quality of stylegan*, *arXiv preprint arXiv:1912.04958* (2019).

[91] S. Marcel and Y. Rodriguez, *Torchvision the machine-vision package of torch*, in *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1485–1488, 2010.

[92] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, *Cnn-generated images are surprisingly easy to spot... for now*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 7, 2020.

[93] Y. Rubner, C. Tomasi, and L. J. Guibas, *A metric for distributions with applications to image databases*, in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pp. 59–66, IEEE, 1998.

[94] M. Kirchner, *Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue*, in *Proceedings of the 10th ACM workshop on Multimedia and security*, pp. 11–20, 2008.

[95] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, *arXiv preprint arXiv:1706.06083* (2017).

[96] G. W. Ding, L. Wang, and X. Jin, *AdverTorch v0.1: An adversarial robustness toolbox based on pytorch*, *arXiv preprint arXiv:1902.07623* (2019).

[97] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, Ieee, 2009.

[98] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, *The cityscapes dataset for semantic urban scene understanding*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[99] Z. Liu, P. Luo, X. Wang, and X. Tang, *Deep learning face attributes in the wild*, in *Proceedings of International Conference on Computer Vision (ICCV)*, December, 2015.

[100] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, *Semantic image synthesis with spatially-adaptive normalization*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[101] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, *Semantic understanding of scenes through the ade20k dataset*, *International Journal of Computer Vision* **127** (2019), no. 3 302–321.

[102] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, *Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop*, *arXiv preprint arXiv:1506.03365* (2015).

[103] M. Barni, K. Kallas, E. Nowroozi, and B. Tondi, *Cnn detection of gan-generated face images based on cross-band co-occurrences analysis*, *arXiv preprint arXiv:2007.12909* (2020).

[104] C. Gudavalli, M. Goebel, T. Nanjundaswamy, L. Nataraj, S. Chandrasekaran, and B. S. Manjunath, *Resampling estimation based rpc metadata verification in satellite imagery*, *IEEE Transactions on Information Forensics and Security* **18** (2023) 3212–3221.

[105] J. Li, X. Li, B. Yang, and X. Sun, *Segmentation-based image copy-move forgery detection scheme*, *IEEE Transactions on Information Forensics and Security* **10** (2015), no. 3 507–518.

[106] J. Horváth, D. Güera, S. K. Yarlagadda, P. Bestagini, F. M. Zhu, S. Tubaro, and E. J. Delp, *Anomaly-based manipulation detection in satellite images*, *networks* **29** (2019) 21.

[107] C. Gudavalli, E. Rosten, L. Nataraj, S. Chandrasekaran, and B. S. Manjunath, *Seetheseams: Localized detection of seam carving based image forgery in satellite imagery*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1–11, June, 2022.

[108] A. C. Popescu and H. Farid, *Exposing digital forgeries by detecting traces of resampling*, *IEEE Transactions on signal processing* **53** (2005), no. 2 758–767.

[109] B. Mahdian and S. Saic, *Blind authentication using periodic properties of interpolation*, *IEEE Transactions on Information Forensics and Security* **3** (2008), no. 3 529–538.

[110] X. Feng, I. J. Cox, and G. Doerr, *Normalized energy density-based forensic detection of resampled images*, *IEEE Transactions on Multimedia* **14** (2012), no. 3 536–545.

[111] A. Flenner, L. A. Peterson, J. Bunk, T. M. Mohammed, L. Nataraj, and B. Manjunath, *Resampling forgery detection using deep learning and a-contrario analysis*, in *Media Watermarking, Security, and Forensics*, IS&T International Symposium on Electronic Imaging, 2018.

[112] P. Boccardo, E. B. Mondino, F. G. Tonolo, and A. Lingua, *Orthorectification of high resolution satellite images*, *Politecnico di Torino, Dipartimento di Georisorse e Territorio, Torino–Italy* (2004).

[113] A. Popescu and H. Farid, *Exposing digital forgeries by detecting traces of resampling*, *IEEE Transactions on Signal Processing* **53** (2005), no. 2 758–767.

[114] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, *et. al.*, *Image quality assessment: from error visibility to structural similarity*, *IEEE transactions on image processing* **13** (2004), no. 4 600–612.

[115] GeoEye, "Usgs eros archive - commercial satellites - orbview 3." `https://doi.org/10.5066/F7J38R0R`, (accessed May 1, 2023).

[116] USGS, "Usgs earthexplorer." `https://earthexplorer.usgs.gov/`, (accessed May 1, 2023).

[117] E. R. Observation and S. E. Center, "Usgs eros archive - digital elevation - shuttle radar topography mission (srtm) 1 arc-second global." `https://doi.org/10.5066/F7PR7TFT`, (accessed May 1, 2023).

[118] CVAT.ai Corporation, *Computer Vision Annotation Tool (CVAT)*, Sept., 2022.

[119] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, *Segment anything*, *arXiv:2304.02643* (2023).

[120] E. Prashnani, M. Goebel, and B. Manjunath, *Generalizable deepfake detection with phase-based motion analysis*, *arXiv preprint arXiv:2211.09363* (2022).

[121] Y. Zheng, J. Bao, D. Chen, M. Zeng, and F. Wen, *Exploring temporal coherence for more general video face forgery detection*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[122] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, *Faceforensics++: Learning to detect manipulated facial images*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[123] H. Zhao, W. Zhou, D. Chen, T. Wei, W. Zhang, and N. Yu, *Multi-attentional deepfake detection*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2185–2194, 2021.

[124] L. Chai, D. Bau, S.-N. Lim, and P. Isola, *What makes fake images detectable? understanding properties that generalize*, in *ECCV*, 2020.

[125] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, and B. Guo, *Face x-ray for more general face forgery detection*, in *CVPR*, 2020.

[126] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, *Recurrent convolutional strategies for face manipulation detection in videos*, *Interfaces (GUI)* (2019).

[127] I. Masi, A. Killekar, R. M. Mascarenhas, S. P. Gurudatt, and W. AbdAlmageed, *Two-branch recurrent network for isolating deepfakes in videos*, in *ECCV*, 2020.

[128] Y. Li and S. Lyu, *Exposing deepfake videos by detecting face warping artifacts*, in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[129] A. Haliassos, K. Vougioukas, S. Petridis, and M. Pantic, *Lips don't lie: A generalisable and robust approach to face forgery detection*, in *CVPR*, 2021.