USC-SIPI REPORT #462

A Green Learning Approach to Image Forensics: Methodology, Applications, and Performance Evaluation

by

Yao Zhu

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL AND COMPUTER ENGINEERING)

August 2023

Acknowledgements

I would like to thank my advisor, C.-C. Jay Kuo, for his guidance and support in every stage of my PhD studies. I have enjoyed every moment working with him. His wisdom, his spirit and his persistence towards the truth will always be my guidance in the future.

I would like to also express my appreciation to the committee members, Antonio Ortega, Justin Haldar, Suya You and Jernej Barbic for their time and helpful advice.

I would like to thank all of the lab members in MCL, working, studying and chilling with them have been a precious and memorable time in my PhD life.

Finally, I would like to thank my family, who always be there for me in my ups and downs. I'm not able to make this far without their wholehearted love.

Contents

List of Tables

List of Figures

Abstract

Fake multimedia has become a central problem in the last few years, especially after the advent of neural networks. Fake multimedia are usually created by whole generation, partial tampering or information hiding. Media forensics, on the contrary, aims to detect the fake contents or discover the hidden information from fake objects. It leverages the fact that manipulation actions leave detectable traces, making fake media objects statistically distinguishable from genuine ones.

In this dissertation, we specifically study two long-standing problems in image forensics: GAN-generated image detection and spatial image steganalysis. The former one aims to detect images that are synthesized by generative models. The latter one focus on distinguishing stego and cover images in spatial domain, where stego images are generated by various content-adaptive steganography algorithms. The stego signal that are embedded into cover images is so weak that the difference in pixel domain in only +1 or -1.

Existing GAN-generated image detection methods are all based on deep neural networks. However, they often need enormous amount of data or intensive data augmentation to maintain its performance with respect to unseen dataset. This motivates us to find a green (light-weight), robust and high-performance GAN-fake image detector. We propose RGGID, which utilizes the assumption that generative models often fail to synthesize well on high-quality details or complex texture regions. We make decision on blocks from those regions and select discriminant soft scores for image-wise decision fusion. RGGID offers a green solution for GAN-generated image detector since its model size is significantly smaller than

that of deep neural networks (DNNs). We apply common manipulations to real/fake source images, including JPEG compression, resizing and Gaussian additive noise, and demonstrate the robustness of RGGID to these manipulations. Furthermore, we prove the generalization ability of RGGID on 11 unseen generative architecture and dataset by training soley on ProGAN and test on other dataset.

Compared to GAN-fake image detection, image steganalysis is a more challenging task. There exist traditional method and deep learning-based method for steganalysis. CNN-based models are proved to have better performance than the three-step traditional machine learning method. However, more secure and complicated steganography schemes force CNN architectures to go deeper and denser, which inevitably results in the insatiable need of memory and computational resources. Motivated by the disadvantages of both methods, we proposed a novel learning solution to image steganalysis based on the green learning paradigm, called Green Steganalyzer (GS). GS consists of three modules: 1) pixel-based anomaly prediction, 2) embedding location detection, and 3) decision fusion for image-level detection. In the first module, GS decomposes an image into patches, adopts Saab transforms for feature extraction, and conducts self-supervised learning to predict an anomaly score of their center pixel. In the second module, GS analyzes the anomaly scores of a pixel and its neighborhood to find pixels of higher embedding probabilities. In the third module, GS focuses on pixels of higher embedding probabilities and fuses their anomaly scores to make final image-level classification. Compared with state-of-the-art deep-learning models, GS achieves comparable detection performance against S-UNIWARD, WOW and HILL steganography schemes with significantly lower computational complexity and a smaller model size, making it attractive for mobile/edge applications. Furthermore, GS is mathematically transparent because of its modular design supported by logical arguments.

Chapter 1

Introduction

1.1 Significance of the Research

Recent years, rapid advances of image generation and manipulation techniques have pushed manipulated content into a higher realism level. The boundary between real and synthetic image are largely narrowed. On the one hand, it provides a new horizon to a series of exciting applications such as creative arts, film production and game design. On the other hand, enormous security threats from manipulated content are inevitably posed upon us. As the prevalence of image/video editing software on social media, any individual can create realistic fake images and videos freely. In addition, with the help of deep-learning tools like Generative Adversarial Networks (GAN) and AutoEncoders, creating realistic manipulated images or videos is easy as long as one can have access to large amount of data. Figure 1.1 shows some popular deepfake manipulations from YouTube where manipulated videos (top row) are synthesized from original ones (bottom row). These videos are made for fun and manipulations are easy to spot. However, similar deepfake videos may be used maliciously, especially for political purposes. By altering identity or information from the original image/video, manipulated image/video can easily mislead the public, resulting in unconvincing journalism, even unconvincing government.

Figure 1.1: Examples of deepfake manipulations from YouTube. Top: manipulated video frames. Bottom: original video frames.

Figure 1.2: Examples of fake faces from https://this-person-does-not-exist.com.

Similar example of fully-synthetic face images are showed in Figure 1.2, which are randomly acquired from https://this-person-does-not-exist.com. The fake faces are generated by StyleGAN [43], a neural-network based generative model which works on style transfer of human faces, such as gender alternation, hair style transfer, etc. The generated faces are too realistic so that it cannot be distinguished by human eyes. If anyone falsify an identity using fake images like Figure 1.2, social credit will be posed under serious threat. Under this circumstances, it's significant to develop a reliable fake-image detector for automatically detecting fake images.

Other than GAN-fake image detection, a more challenging task in image forensics is steganalysis. Steganalysis aims to detect the media which have secret information embedded, while steganography wants to embed secret information as much as possible without being detected by steganlyzers. Steganography is often used in encrypted communication, especially during military operation. Cover medium used by steganography include image,

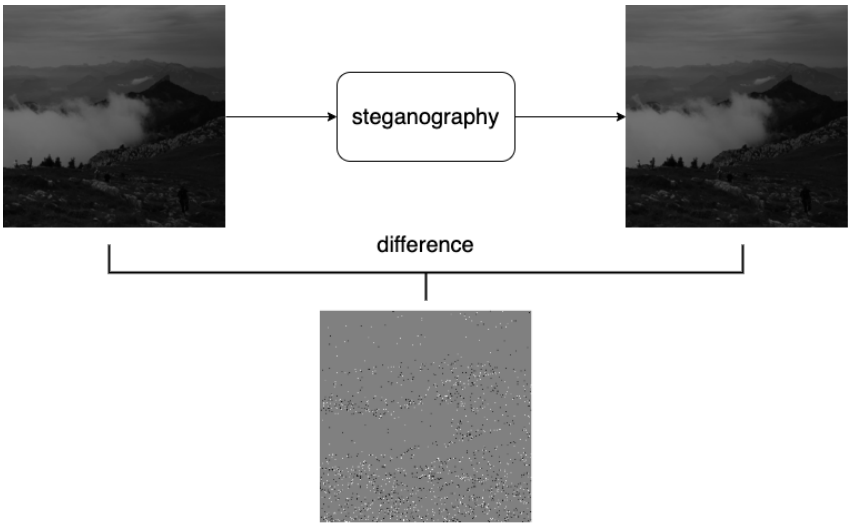


Figure 1.3: Example of cover (left) and stego (right) image. Difference map between cover and stego image is shown below.

audio, video or even text. In this dissertation, we only focus on image steganalysis. Figure 1.3 presents an example of embedded image (stego) and original image (cover). A difference map between example cover and stego image is also presented in Figure 1.3. Visually, we cannot see any difference between the two, since the alternation in pixel domain is only ± 1. Steganalysis has been a long-going topic in image forensics. In early stages, embedded pixels are pervasive in terms of embedding location across the whole image. One of the famous steganography scheme of this kind is LSB replacement and LSB matching, where LSB stands for Least Significant Bit. However, this steganography technique is not secure enough as steganalysis methods evolve. With the advances of steganography techniques, embedded stego-signals become more challenging to be detected, as they are more concentrated in complex texture regions. In this case, it's necessary to develop a steganalysis solution for content-adaptive steganography schemes.

In this dissertation, we first focus on GAN-generated image detection. Then, we move on to image steganalysis, which is to detect the weak noise-like embedded signal from images. For both tasks, we aim to provide light-weight, mathematically transparent and robust

solutions. While we are able to achieve higher or comparable performance with CNN-based methods, our model size and computational cost are significantly less.

1.2 Contributions of the Research

1.2.1 Robust and Green GAN-fake Image Detector

For GAN-generated image detection, there exist several deep learning-based models to solve this problem. Existing CNN-based models usually have high performance under specific dataset or generative architecture. However, when a new generative model or dataset comes out, CNN-based models often fail to generalize well on it. Even if they can maintain performance, enormous amount of data or intensive data augmentation are needed when training models. To address this problem, we propose a light-weight, robust, and mathematically transparent solution for GAN-fake image detection.

- We take advantage of the up-sampling architecture in GAN-based generative models. That is, they cannot synthesize well on high-frequency components of images, such as high-quality details, complex textures and edges. We decompose image into several small image blocks, and select blocks from images that are from aforementioned complex texture regions since they contain more discriminant high-frequency components.

- We incorporate unsupervised feature learning method called PixelHop to extract features from selected blocks. In PixelHop unit, filter weights are derived from a variant of PCA transform without any end-to-end training or back-propagation. The computational complexity and space complexity are largely reduced because of it.

- We propose to make decision on each feature map of block-wise features. By analyzing classifier performance on each feature map, we collect soft decision scores from discriminant feature maps, which mainly come from high frequency channels.

- We propose to use two-end ensemble strategy in decision fusion. Since collected soft decision scores form a distribution and real and fake block scores are more separable in tail regions, we sample small percentage (10%) of scores from the two-end of the distribution and form a feature vector for ensemble classifier. In this case, feature dimension for each image is fixed for training ensemble classifier.

- Other than evaluating our method on raw images, we design 3 image manipulation scenarios, JPEG compression, image resize and additive noise to testify the robustness of our method against various image manipulation. We analyze the effect of image manipulation on the fundamental assumption of our method. Robustness of our method is demonstrated by the maintained performance under all 3 manipulation cases.

- We demonstrate the generalization ability of our method on unseen generative architectures and dataset by training on solely ProGAN images and test on other 11 various generative models, including 6 GAN-based models, 4 CNN and 1 autoencoder based model. Superior performance validates our method can be well generalized to all state-of-the-art unseen dataset.

1.2.2 Green Steganalysis

There exist traditional and deep learning-based methods for image steganalysis. For traditional methods, hand-crafted filters are often utilized to suppress image content and increase SNR of stego-signal. With the increasing security of steganography techniques, heuristically designed filters are not capable enough to describe the complex embedding situations. Deep-learning-based methods usually have better performance than traditional ones since the system is unified and optimized in end-to-end manner. However, they suffer from exhausting need of computational source to fulfill the task. To address the problem, we propose a green steganalysis solution (GS) that does not utilize hand-crafted filters or end-to-end training.

- We take advantage of content-adaptive steganography algorithm characteristic such that, less pixels are altered in smooth regions. We decompose image into small patches and partition them into several groups, where patches in each group share similar content. In this case, filters learned from each group will be more concentrated on 'stego-signal' instead of image content.

- We incorporate unsupervised feature learning method called Saab transform to learn content-adaptive filters and extract features from each group. Saab filter coefficients are derived from Saab transform, a variant of PCA transform without any end-to-end training or back-propagation. Because of this, computational complexity and space complexity are largely reduced.

- We assign anomaly scores on patches according to anomaly detection classifier and analyze the anomaly score map. We notice that difference between cover and stego anomaly score maps are more noticeable around embedded locations. Decision fusion of anomaly scores from embedded locations have great potential in image-wise classification.

- We design an embed location detector to detect possible embed location. We utilize matched filter to ensure the robustness and enhance the detectability of embed location detector, where matched filter is the element-wise average of anomaly scores in a certain neighborhood of embedding location.

- We directly use anomaly scores from detected possible embed locations as image-wise feature. We experiment various image feature dimension, train image-level classifier for each of them, and get the final decision by simple decision fusion.

- Performance comparison between our method and other state-of-the-art method under various payload and different steganography algorithms demonstrate the high-performance of our method. We prove the 'green' characteristic by comparing number of parameters and number of FLOPs between our method and state-of-the-art neural network-based models, where our model size rank the second smallest.

1.3 Organization of the dissertation

The rest of the dissertation is organized as follows. We first review research background of image forensics, GAN-generated image detection, steganography and steganalysis in Chapter 2. In Chapter 3, we propose a green, robust GAN-fake image detection method based on the fundamental assumption that GAN architectures often fail to synthesize well on high-frequency components of images. In Chapter 4, we propose a green image steganalysis solution where no heuristically designed filters or end-to-end training is needed. Finally, we concluding remarks and future research directions are given in Chapter 5.

Chapter 2

Research Background

2.1 Image Forensics

Image Forensics is the practice of collecting, analyzing, and reporting on digital evidence so that it is admissible in the court. Image forensics is challenging to forensic investigators such that it uses a mixture of techniques to create and conceal the intended content. Thanks to the wide adoption of mobile devices, cheaper storage, high bandwidth, people are generating a enormous amount of data on social media every day. This growth has pushed the development of image forensics as well. Also, neural networks open a new door of analytic tool for forensic experts to surpass the capability of individuals and effectively analyze and process the data.

Image forensics involves the set of techniques used for the analysis of the authenticity and integrity of images. It aims to reveal the history of digital content, identifying the acquisition device that produced the data, validating the integrity of the contents and retrieving information from image content.

Image Forensics divides its wide coverage into 2 main categories – Passive Image Authentication and Active Image Authentication. Figure 2.1 illustrates the categories of image

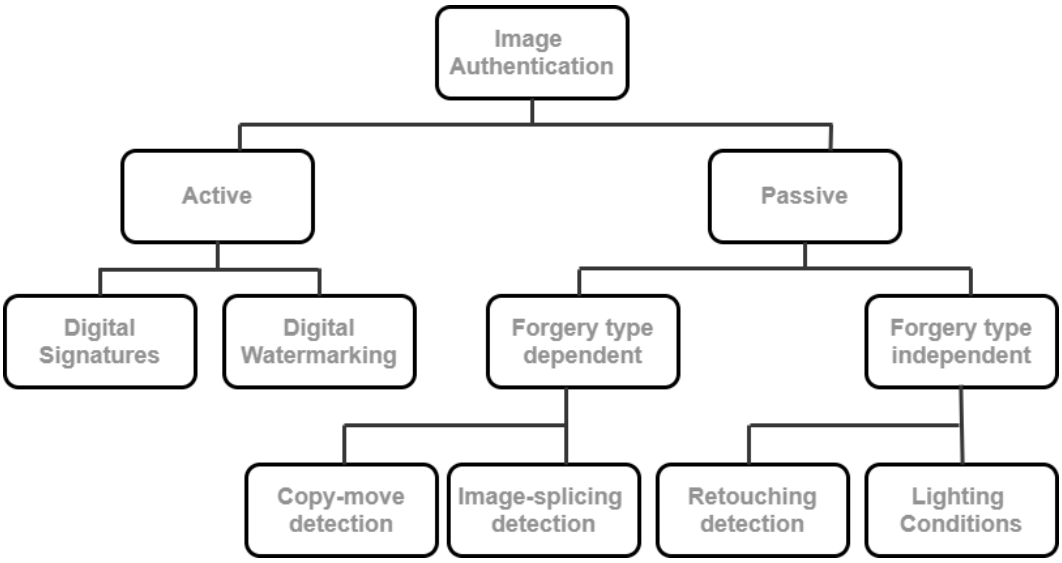


Figure 2.1: Hierarchy structure of image forensics

forensics and its typical tasks. For Passive authentication, it is also known as image forensics. It uses only image with no prior knowledge for accessing the integrity of the image. Passive authentication works on the assumption that even though tampering with the image may not leave any visual trace but they are likely to alter the underlying statistics. This means that image forgeries may disturb the underlying properties of the image, quality of the image, even though human eyes cannot differentiate the forgery clue. Popular tasks which belong to this categories are: image splicing localization, copy-move detection and image in-painting detection. Our first work, GAN-generated image detection, also belongs to this kind. In Active Image Authentication, a known authentication code is embedded in the image at the time of image generation. Image with authentication code are sent to receiving end for accessing its integrity. Verification of the code authenticates the originality of the image. The second work in this dissertation, image steganalysis, belongs to this category.

There are also other fields that are widely explored in image forensics. They include:

1. Source identification

In this field, image forensics is used to determine the source of a particular image file, including identifying the camera or device used to capture an image. This involves analyzing specific characteristics such as sensor noise patterns, lens aberrations, and other unique signatures left by the capture device.

2. Authenticity verification

   In this field, forensics techniques verifies the authenticity of digital image by examining metadata, such as timestamps, geo-location information, and digital signatures. It also involves analyzing compression artifacts, noise patterns, and inconsistencies in the file structure to assess whether the image file has been modified or manipulated.

3. Format and compression analysis

   Forensic researchers are interested in investigating the file format and compression techniques used in digital images. By examining the compression artifacts and analyzing the file structure, it is possible to gain insights into the history and authenticity of the image file.

4. Audio and video authentication

   Other than images, the analysis of audio and video recordings is also vital in forensics field to determine their authenticity and integrity. This includes identifying audio manipulations, detecting voice alterations, analyzing speech patterns, and examining video synchronization and temporal inconsistencies.

Image forensics plays a vital role in various domains, including law enforcement, journalism, intelligence agencies, digital rights management, copyright protection, and ensuring the integrity of digital evidence in legal proceedings. By employing scientific methodologies and advanced algorithms, image forensics contributes to the verification and authentication

of digital images, enhancing our ability to discern between genuine and manipulated images in an increasingly digital world.

2.2 GAN-generated Image Detection

Modern image generation models are built upon Generative Adversarial Networks (GANs) [33]. CycleGAN [37] is a well-known GAN model that can change the image style, switch semantic objects and translate images from one domain to a different domain without paired training images. GauGAN [72] can translate human sketches to photo-realistic images. One common application of style transfer models is face manipulation. For example, StarGAN [18] can change the expression of a face, alter hair style, or modify skin color. StyleGAN [43] generates fully synthetic human faces with specific high-level attributes such as poses or identities. ProGAN [42] synthesizes high-resolution high-variation face images by progressively growing both the generator and discriminator. BigGAN [8] aims at generating high-quality high-resolution images by leveraging a sequence of best practices on training class-conditional images and scaling up batch sizes.

GAN artifacts have been carefully studied and exploited in GAN-fake image detection. One type of artifact results from the convolutional up-sampling structure of neural networks. Another kind of artifact appears in the form of color distortion, which was used to capture the dissonant or asymmetric characteristics of images in [56,69]. Another source of artifacts arises from the artificial fingerprint associated with a GAN architecture. The persistence of these fingerprints across different GAN models, datasets and resolutions was studied in [98]. A GAN simulator, called AutoGAN, was introduced in [108] to simulate artifacts of popular GAN models. The authors of [108] identified an artifact that manifests itself as spectral peaks in the frequency domain, and thus proposed feeding the spectral-domain input to a classifier for GAN-fake image detection.

Several neural networks have been proposed for GAN-fake image detection. Nataraj *et al.* [71] used the co-occurrence matrix to derive hand-crafted features and fed them to a CNN for detection. Inspired by image steganalysis, Cozzolino *et al.* [21] proposed a CNN to mimic rich models [30] in feature extraction and real/fake classification. Recently, Wang *et al.* [86] trained a CNN classifier with a large number of ProGAN-generated images and evaluated it on images synthesized by eleven other GAN models. They showed the effectiveness of extensive data augmentation in improving the generalization ability of a CNN classifier.

Most research on GAN-fake image detectors has been developed and tested on raw real/fake images. However, most real world images do not exist in the raw image domain. They are compressed for ease of storage and transmission. They may be rescaled to fit different screen sizes. Furthermore, an attacker may add Gaussian noise to real/fake images to make their differentiation more challenging. There is much less work on the robustness of GAN-fake image detectors against image manipulations. Marra *et al.* [68] compared the performance of multiple neural networks under Twitter's compression. They also considered the compression setting mismatch between training and testing datasets to evaluate the robustness of CNN-based classifiers. Wang *et al.* [86] explored data augmentation to enhance the robustness of a detector.

2.3 Image Steganography

In this section, we introduce some background knowledge of steganography. It includes the formal definition of steganography, early steganography algorithms such as Least Significant Bit (LSB), and recent content adaptive steganography algorithms.

2.3.1 Definition

Steganography is a way of covert communication. Instead of communicating the actual message, it hides or embeds the message in another object, and communicate the altered object in via secret channel. In steganographic literature, it is often refered as steganographic channel. The communication include three parties, sender and receiver, while a potential eavesdropper also exists.

To ensure the security of their communication, the sender and receiver share a set of secret keys $\mathbf{k} \in \mathcal{K}$ in advance. They can now establish a steganograpgic system that consists of various components: a cover source $\{\mathcal{C}, P^{(c)}\}$, a message source $\mathcal{M}, P^{(m)}$, a set of stego keys $\mathcal{K}$, and embedding and extracting functions Emb and Ext. The cover source contains all possible cover objects $x \in \mathcal{C}$ and their respective distribution $P^{(c)}$. Similarly, the message source is comprised of all possible messages $\mathbf{m} \in \mathcal{M}$ and their distribution $P^{(m)}$. $\mathcal{S}$ is denoted as the set of all potential stego objects with distribution $P^{(s)}$. The embedding function takes a cover object, the message to be conveyed, and the shared secret key,

$$Emb : \mathcal{C} \times \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{S} \tag{2.1}$$

and generates a stego object that carries the message.

$$\mathbf{y} = Emb(\mathbf{x}, \mathbf{m}, \mathbf{k}) \in \mathcal{S} \tag{2.2}$$

Conversely, the extraction function processes the stego object

$$Ext : \mathcal{S} \times \mathcal{K} \rightarrow \mathcal{M}) \tag{2.3}$$

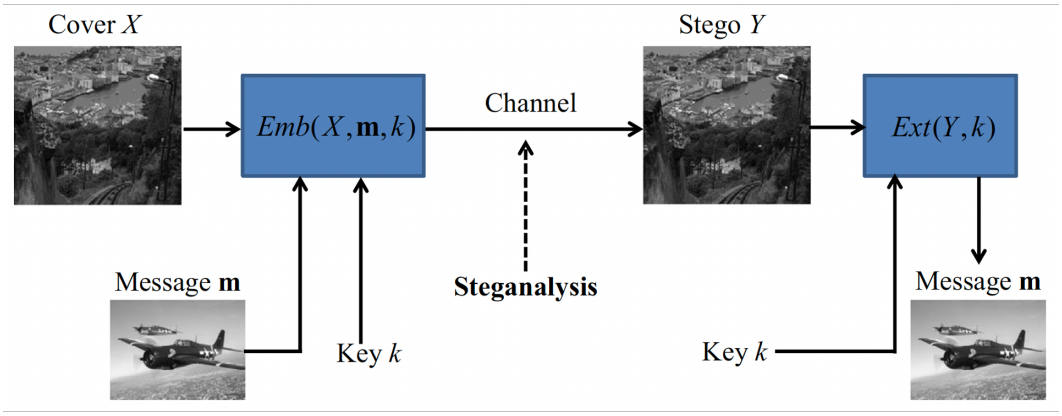


Figure 2.2: Steganographic channel.

and extracts the secret message for all cover objects $\mathbf{x} \in \mathcal{C}$, messages $\mathbf{m} \in \mathcal{M}$, and secret keys $\mathbf{k} \in \mathcal{K}$.

$$m = Ext(\mathbf{x}, \mathbf{m}, \mathbf{k}), \mathbf{k}) \tag{2.4}$$

A visualization of the steganographic system above is shown in Figure 2.2.

The security of the steganographic channel is also critical. Considering the cover source denoted above $\{\mathcal{C}, P^{(c)}\}$, we can consider any cover object as an observation of a random variable that follows the cover distribution $X \sim P^{(c)}$. Similarly, a stego object can be viewed as an observation of a random variable that follows the stego distribution $Y \sim P^{(s)}$. The steganographic system is only secure when the cover and stego distributions are statistically indistinguishable. The mathematical evaluation of 'how distinguishable are the two distribution' is to measure the dissimilarity between them. One metric is Kullback-Leibler divergence (KL divergence or relative entropy) from information theory.

$$D_{KL}(P^{(c)}||P^{(s)}) = \sum_{\mathbf{x}\in\mathcal{C}} P^{(c)}(\mathbf{x}) log \frac{P^{(c)}(\mathbf{x})}{P^{(s)}(\mathbf{x})} \tag{2.5}$$

The stego system is considered perfectly secure (undetectable) when $D_{KL}(P^{(c)}||P^{(s)}) = 0$, indicating that the distributions $P^{(c)}$ and $P^{(s)}$ are identical, thereby making it impossible for

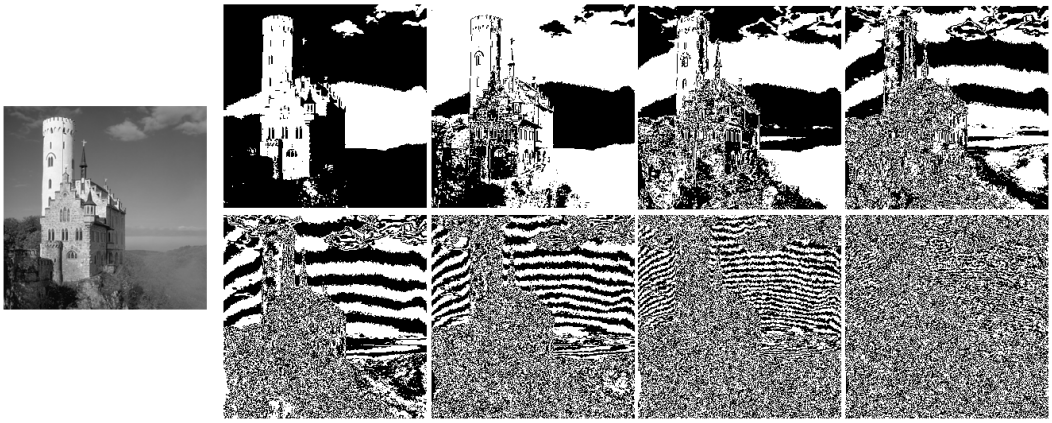


Figure 2.3: An exampler image (left) and its 8 bitplanes, from most significant (top left) to least significant (bottom right) in raster order.

the eavesdropper to differentiate between cover and stego objects. Although achieving this level of security is desirable, it is challenging in practice. Hence, the security requirements are somewhat loosened. We say a stegano-system is ε-secure if $D_{KL}(P^{(c)}||P^{(s)}) \leq \varepsilon$.

2.3.2 Early Steganography Algorithms

Among the decades history of image steganography, early algorithms hide the messages within the Least Significant Bits (LSB) of pixel values. Figure 2.3 shows an example of bitplanes from its Wikipedia. Among the 8 bitplanes, top left depicts the most significant bitplane, which preserves most of the image content. The bottom right depicts the least significant bitplane, which is almost noise. The noise pattern observed in the LSB plane stems from various noise sources within digital imaging sensors, including shot noise and electronic noise. However, this noise prevents us to hide information within computer-generated images because they don't have such noise characteristics.

The LSB steganography algorithms were once considered secure due to the noise-like appearance it yielded. One of the simplest algorithms utilizing the LSB plane is LSB Replacement (LSBR). This algorithm employs a secret stego key to establish a pseudo-random path across cover pixels, embedding message bits into their LSB values. LSBR is

a typical example of non-adaptive steganography, as the potential embedding changes can be pervasive across the whole image. Despite LSBR resembles random noises visually, the LSB plane is not entirely random. Numerous powerful steganalysis algorithms have been developed against this steganography [28, 29, 45].

An alternative form of the embedding algorithm is known as LSB Matching (LSBM). It involves matching the least significant bit (LSB) of the cover to a message bit by randomly adjusting the pixel value by +1 or -1 when the LSB doesn't carry the message bit. Unlike LSBR, LSBM doesn't introduce any noticeable artifacts into the histogram despite the potential alteration of multiple bits in the pixel's binary representation. Additionally, LSBM maintains the pixel mean while changing its variance, which is different from LSBR that modifies the pixel mean. Detecting changes in pixel mean is a simpler denoising task compared to estimating variance, making LSBR more easily to be detected. Due to these reasons, LSBM can be utilized for content-adaptive embedding. However, in its non-adaptive form, there are still several accurate and straightforward statistical steganalysis methods [19, 44, 59].

Up to now, we haven't introduced how the message is actually been embedded. We only introduced how steganography algorithms find the possible location to do embedding. Once the embedding location preference have been set, message will be coded and embedded in cover images. The coding techniques in steganography is motivated by error-correcting codes. One of the latest development of error-correcting codes is Syndrome Trellis Codes (STCs) [27]. It minimizes the distortion between cover and stego and achieves nearly optimal performance on Rate-Distortion Bound. The rate distortion bound is the lower bound of change rate, i.e. the average distortion per pixel.

$$\beta \geq H_3^{-1}(\alpha) \tag{2.6}$$

where H_3^{-1} is the inverse of ternary (+1, -1, 0) entropy function and α is the payload or message length. In this dissertation, we assume that STCs can achieve the optimal coding given by the RD bound. In this case, change rates will in turn have the following condition

$$\sum_{i=1}^{n} H_3(\beta_i) = n\alpha \tag{2.7}$$

where β_i represents the change rate for pixel i. In content-adaptive steganography, change rate can be different for different locations in cover image. Given certain image size and message length, RHS of equation 2.7 is deterministic, thus, researchers put more efforts in the calculation of β_i. More details of STCs, especially the derivation from equation 2.6 to 2.7 can be found in its original paper [27].

2.3.3 Content-adaptive Steganography

A significant breakthrough in steganography occurred with the introduction of content-adaptivity. This approach involves incorporating knowledge about the content of the cover image into the embedding scheme. The knowledge of image content ensures that modifications are avoided in parts of the image that are easy to be detected. Intuitively, this approach is reasonal because even a small adjustment, such as +1 or -1, to a pixel in a region that is easily predictable would likely be highly detectable. To achieve this, embedding costs are assigned to different regions of the cover image. Currently, there are three primary approaches to content-adaptive steganography: cost-based, model-based, and adversarial. In the following paragraphs, we will provide a brief explanation of each of these strategies.

1. **cost-based steganography**

   This is the most widely developed content-adaptive steganography strategy. It heuristically design the cost of changing the $i-th$ element in cover image as ρ_i. By saying heuristically design, the costs are typically derived through experimental evaluation as

well as the feedback obtained from state-of-the-art steganalysis detectors. Under the assumption that the cost of each pixel does not interact or influence with each other, the total distortion because of the embedding can be defined as the simple summation of all cost over cover image. There is no cost for no embedding or no change of cover pixel.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \rho_i[x_i \neq y_i] \tag{2.8}$$

Minimizing the total distortion with the constraint in 2.7 can be done by Lagrangian. Since we don't have the prior knowledge that where will the embedding location be, because of the randomness from Syndrome Trellis Codes (STCs), the minimization is analyzed on expected distortion

$$\mathbb{E}[D(\mathbf{x}, \mathbf{y})] = \sum_{i=1}^{n} \rho_i \beta_i \tag{2.9}$$

The expected distortion is usually interchangeable with distortion. From Equation 2.9, the optimal change rate β_i that minimizing the expected distortion can be derived as

$$\beta_i = \frac{e^{-\lambda \rho_i}}{1 + 2e^{-\lambda \rho_i}} \tag{2.10}$$

λ is the lagrangian multiplier, which is decided by the payload α constraint in Equation 2.7.

There has been several successful content adaptive steganography approaches in spatial domain. They include S-UNIWARD [35], WOW [34], HILL [54] and HUGO [26]. The embedding procedures of these approaches are the same, but the analysis of embedding impact or embedding cost are different.

Specifically, in [26], Highly Undetectable steGO (HUGO) method uses weighted difference of feature vector to analyze the distortion of embedding. In [34], Holub and Fridrich *et al.* proposed to analyze the change in directional high-pass filter response after embedding and model the embedding cost in individual pixel as Wavelet Obtained Weights (WOW). They further optimized the embedding cost according to directional residuals in UNIversal WAvelet Relative Distortion (S-UNIWARD) [35], where the residuals are calculated by a filter bank. Proposed by Li *et al.* in [54], HILL method uses one high-pass filter and two low-pass filters to calculate embedding cost. The high-pass filter is to locate the complex (low-cost) regions and the following two low-pass filters aim to make the low-cost regions more congregated.

2. **Model-based steganography**

In recent years, steganographers start to lay eyes on deep learning-based solutions for more complicated information embedding scheme. Theoretically, the steganography/steganalysis problem is like a game between steganographer and steganalyst: each player in the game wants to find a strategy that maximize the winning chance. This coincides with the Generative Adversarial Networks [33] proposed by Goodfellow *et al.*, which also 'simulate' a game between image generator and discriminator. For this, GAN system is widely adopted to steganography. There are 2 categories of this kind: (1) steganography via image synthesis. This type of approach generate cover images from GAN, then hide message in synthetic images (Hu et al. [36]) or simply consider synthetic image as stego (Shi et al. [81]). (2) steganography by generating probability map of modifications. The two innovative examples are ASDL-GAN [84] and UT-6HPF-GAN [92]. In this category, generator network of GAN generates a modification probability map from cover image. The probability map then used to generate the modification map via embedding simulator. The discriminator network

of GAN takes the summation of cover image and modification map as input and aims to detect the stego image from cover one. Different from traditional adaptive steganographic approaches, this family of steganography learns modification probability map in an end-to-end manner.

3. **Adversarial in steganography**

   Besides the two GAN-based steganography categories, there exist two other CNN-based steganography categories: steganography via adversarial embedding and steganography as 3-player game. The former has an example ADV-EMB proposed by Tang *et al.* in [82]. The cost map of cover image is updated iteratively according to the gradient of loss of steganalyzer network. In this case, steganalyzer network is fooled and results in more secure steganography approach. The latter, 3-player game steganography approach, simulates the steganography/steganalysis problem as a game between steganographer (sender), receiver, and steganlyzer. Examples are HiDDeN [113] and SteganoGAN [102], which utilize encoder-decoder architecture to realize information embedding and retrieval. The role of steganalyzer is mimicked by a third-network to improve security.

2.4 Image Steganalysis

In this section, we review several state-of-the-art steganalysis algorithms. We introduce traditional methods such as SRM and CNN-based methods, such as Xu-Net, Ye-Net, Yedroudj-Net,SRN-Net, Zhu-Net and GBRAS-Net.

2.4.1 Traditional Image Steganalysis Methods

Before the emerging of neural networks, steganlyst have already constructed mature techniques to perform steganalysis. Traditional steganalysis methods often use hand-crafted

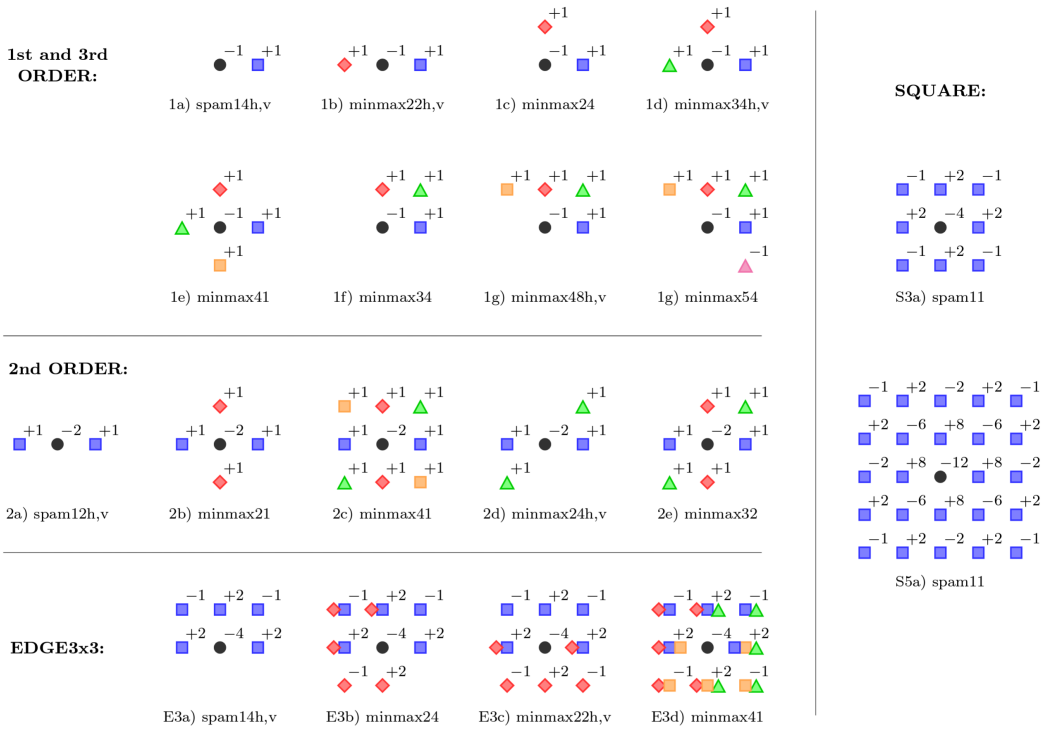


Figure 2.4: SRM kernels from [30].

filters to extract features from images. Then, statistical analysis tools such as histogram or co-occurrence matrix are applied to extract higher-order information of images. At last, machine learning classifiers are adopted to do the final decision.

As pioneers in the field, Fridrich *et al.* introduced Spatial Rich Model(SRM) [30], which combines a set of rich image models (depicted in Figure 2.4) with ensemble SVM classifiers. By considering various diverse relationships between pixels, they constructed the rich models from noise component of images. Features from each rich model are then concatenated to form a long feature vector and ensemble classifiers are used to give final decision. SRM are proved successful against a wide spectrum of steganographic schemes. In [63], Lu *et al.* proposed to use Fisher criterion to do dimension reduction on steganalytic features. They analyzed the separability of single-dimension and multiple dimension spatial-domain features by Euclidean distance, then used Fisher criterion to select the feature components

with best separability as final steganalytic features. In [83], Tang *et al.* proposed to assign different weights to pixels based on their embedding probabilities during feature extraction. By doing so, pixels with higher embedding probabilities are assigned larger weights and thus contribute more to steganalysis. The proposed scheme has been proved effective especially under low embedding rate scenarios (lower than 0.20 bpp).

2.4.2 Deep Learning-based Image Steganalysis Methods

Arise of deep-learning based models have pushed steganalysis performance to the next level. Qian *et al.* were the first to introduce a CNN-based model to the steganalysis battlefield, which is called GNCNN [74]. It is a customized CNN model with predefined high-pass filter as preprocessing layer. Also, it introduced Gaussian function as non-linear activation in convolutional layers. Compared with classical steganalysis methods, GNCNN is the first to automate feature extraction step and classification step in a unified system. It achieved comparable performance with classical methods on three spatial domain steganography approaches under various payload (0.3-0.5 bpp). In [91], Xu *et al.* also used fixed high-pass filter layer to extract noise residual. They took absolute values of feature from 5 groups of convolutional modules and used $Tanh$ as activation function. Ye *et al.* proposed Ye-Net in [94], which utilized filter banks from Spatial Rich Model [30] as initialization of weights in the first convolutional module. They incorporated the knowledge of selection channel into the designed CNN architecture, and developed a novel activation function called Truncated Linear Unit (TLU) to better suit the nature of stego-noise (± 1). In [96], Yedroudj *et al.* proposed a new CNN-based model called Yedroudj-NET, which brings together the merits of its predecessors. They incorporated predefined high-pass filter from SRM [30] as preprocessing step, and bind both Absolute Value activation (ABS) and Truncated Linear Unit (TLU) in convolutional module.

In [89], Wu *et al.* introduced ResNet architecture into steganalysis, which they called DRN. Similar to previous models, it also has a high-pass filter as preprocessing step to force the system quickly converge. They use residual learning blocks to preserve features from weak stego signals. In [7], Boroumand *et al.* proposed SRNet, which is a deep residual paradigm that minimizes the usage of heuristics elements in system. It does not use any predefined high-pass filter as preprocessing module like previous models. It disabled pooling step in the beginning convolutional blocks to prevent the loss of information from weak stego signals. SRNet is currently one of the best approach for high-detection performance. However, it suffers from large model size and computational complexity. In [97], You *et al.* introduced Siamese CNN for steganalysis, which is a novel CNN architecture that has two symmetrical subsets with shared parameters. The proposed network analyze the relationship between the noise components from image sub-regions and make classification based upon it.

2.5 Green Learning

Green learning aims at an energy efficient way to achieve the goal of data-driven learning. The models should have lower training/inference computational complexity, have smaller model sizes, and require fewer training samples while maintaining similar classification or regression performance as deep-learning models. It is desired that their computation can be carried out solely on CPU or small GPU. Thus, green learning solutions are suitable for edge and mobile computing.

Distinct from the end-to-end optimization of deep learning, green learning adopts a modularized design by following the traditional pattern recognition learning paradigm. It consists of "unsupervised feature learning", "supervised feature learning" and "supervised

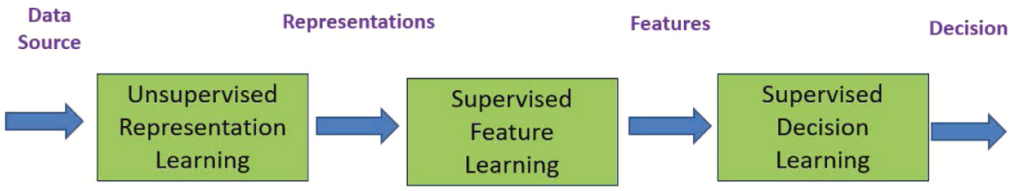


Figure 2.5: An overview of generic Green Learning system from [49].

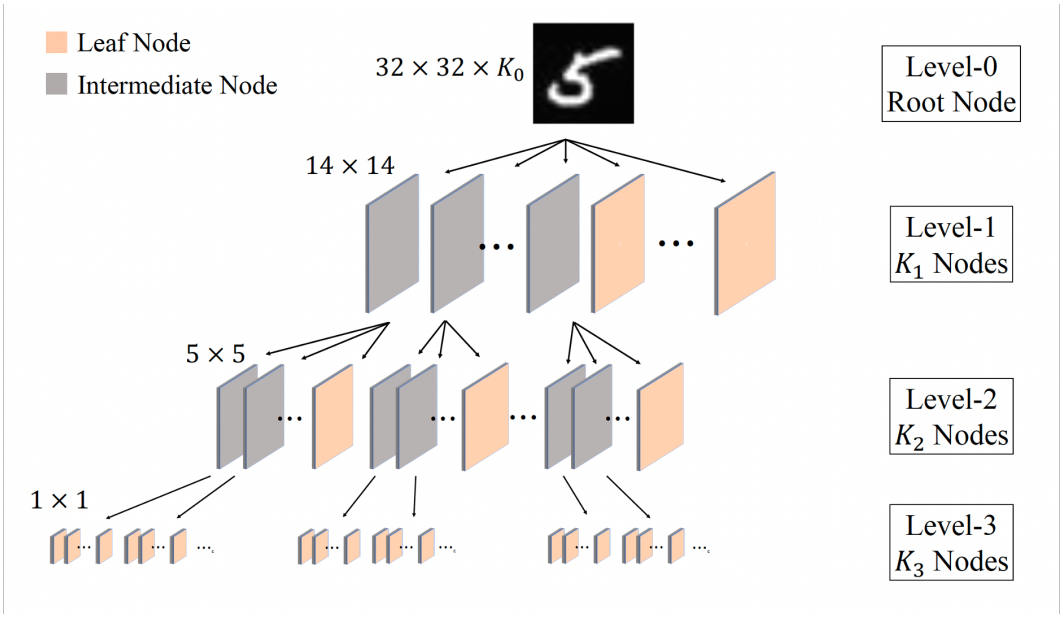


Figure 2.6: Illustration of channel-wise Saab transform from [15].

decision learning", see Figure 2.5. The idea of unsupervised feature learning has been developed in a sequence of papers [14–16, 47, 48, 50]. While filter parameters of CNNs are obtained by back-propagation, filter parameters in green learning are determined by statistical analysis of the neighborhood of a center pixel. Specifically, a variant of Principle Component Analysis (PCA), called the Saab (Subspace approximation via adjusted bias) transform was proposed in [50] to achieve the task. Futhermore, in [15], a variant of Saab transform is proposed, named channel-wise Saab transform. It aims to reduce the computational complexity and space complexity in Saab transform. An illustration of channel-wise Saab transform is showed in Figure 2.6.

Saab transform and Channel-wise Saab transform provide an unsupervised way to derive filter weights in feature learning process. PixelHop, a modularized unit based on Saab transform, which contains both filter learning and feature extraction process, is introduced in [14]. It functions as a convolutional layer in CNN architecture. Similar to PixelHop, PixelHop++ is designed upon channel-wise Saab transform and introduced in [15]. In this dissertation, the two mentioned works both utilized PixelHop module in unsupervised feature learning process. The filters derived from Saab transform are able to extract features from low-frequency component to high-frequency component in images. For image forensics tasks, high-frequency features are usually more advantageous than low-frequency features. Saab transform naturally derive filters that are suitable for various tasks, such as object classification, fake-image detection, etc.

Other than "unsupervised feature learning" from Saab transform, another important aspect of Green learning is "supervised feature learning", that is to use labels/targets to select a subset of features that can achieve similar or even better performance than the whole feature set. The effective utilization of labels or supervision to enhance the performance of a learning system is a crucial aspect in machine learning (ML). Traditional ML approaches primarily employ labels in classifier design, whereas deep learning (DL) goes a step further by utilizing labels to adjust filter weights in both the feature subnet and the decision subnet, leading to improved performance.

In the literature of semi-supervised and supervised feature selection methods, existing techniques can be categorized into three classes: wrapper, filter, and embedded methods. Wrapper methods involve creating multiple models with different subsets of input features and selecting the model that demonstrates the best performance based on the features utilized. Recursive feature elimination (RFE) is an example of a wrapper method. However, this process can be computationally expensive. Filter methods, on the other hand, assess the relationship between input and target variables using statistical measures and select the

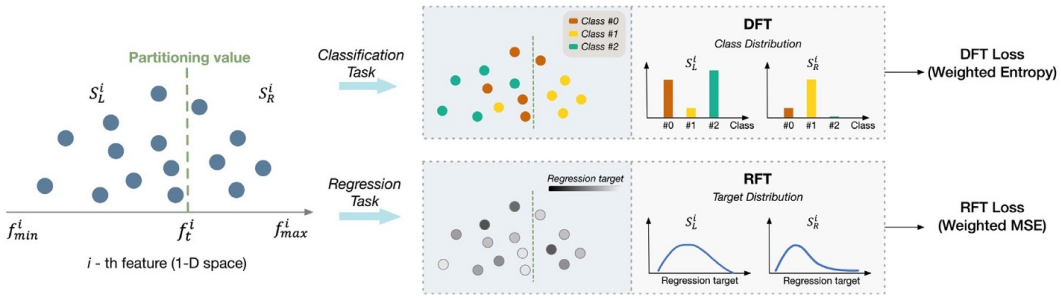


Figure 2.7: Block diagrams of two supervised feature selection methods: the discriminant feature test (DFT) and the relevant feature test (RFT) [93].

variables that exhibit the strongest association with the target variables. Analysis of variance (ANOVA) is one such filter method that efficiently identifies relevant variables with robust performance. Embedded methods, as the name suggests, perform feature selection during the training process and are typically specific to a particular learner. A prominent example is the determination of "feature importance" (FI) obtained from the training procedure of the XGBoost classifier/regressor, also referred to as "feature selection from model" (SFM). Additionally, in [93], the discriminant feature test (DFT) and the relevant feature test (RFT) were introduced for classification and regression problems, respectively, belonging to the filter class.

Figure 2.7 presents the design of supervised feature selection in classification case: DFT and regression case: RFT. Specifically, for the $i-th$ feature, which is a distribution in 1-D space, DFT measures the class distribution in S_L^i and S_R^i to compute the weighted entropy as the DFT loss. Similarly, RFT measures the weighted MSE in both sets as the RFT loss. They are used to select discriminant/relevant features from a large set of representations learned from the source without labels.

The last essential module in green learning is "supervised decision learning". Traditionally, in machine learning applications, machine learning classifiers or regressors are applied in this stage. In green learning, we further explore the potential of machine learning models by "ensemble". It includes the ensemble of features, and the ensemble of machine learning

classifiers. In terms of feature ensemble, green learning make use of the parallel design of PixelHops. Feature derived from different filter sizes are aggregated together for the decision making. The parallel design can not only enrich the receptive fields in feature extraction process, but also increase the robustness of features.

The ensemble of machine learning models is a well-developed area. It has two categories, boosting and bagging. In this dissertation, we use XGBoost classifier [13] in the supervised decision learning stage, which belongs to boosting category. Boosting make use of the errors from previous learners, thus its learning is sequential. Bagging, on the other hand, trains weak learners individually. Their final decision is the majority vote (for classification) or average (for regression) of decisions from all weak learners. A typical example of Bagging is Random Forest. For green learning paradigm, we developed our own boosting and bagging learner, called SLM boost and SLM forest [31].

Green learning methodology has been successfully applied to various computer vision tasks such as image classification [14, 15] and 3D point cloud [38, 40, 103, 105, 106], texture synthesis [52, 53, 99, 100], graph [61, 90] and others [32, 39, 66, 70, 77–79, 85, 87, 88, 93, 101, 104, 109, 110]. In the area of image forensics, green learning solutions have been developed for deepfake video detection [10] and GAN-fake image detection [115].

Chapter 3

RGGID: A Robust and Green GAN-Fake Image Detector

3.1 Introduction

We have witnessed the rapid development of image generation techniques based on convolutional neural networks (CNNs) in general and generative adversarial networks (GANs) [33] in particular. Various GANs have been developed to yield high quality image synthesis and translation performance. The quality of these generated images is so good that it is difficult to distinguish them from real images. This poses a threat to image authenticity and may contribute to a source of dis/misinformation in our society. Effective detection of GAN-fake images has received a lot of attention in recent years.

The challenges of GAN-fake image detection lie in two aspects. First, it is common to apply manipulations to real/fake images in real-world application scenarios. They include JPEG compression, resizing, Gaussian additive noise, etc. The distortions introduced by manipulations may mask small differences between real and fake images and make it even more difficult to perform fake image detection. Thus, it is essential to develop a robust GAN-fake image detector. Second, most state-of-the-art GAN-fake image detectors are built upon deep neural networks (DNNs). They offer good detection performance at the

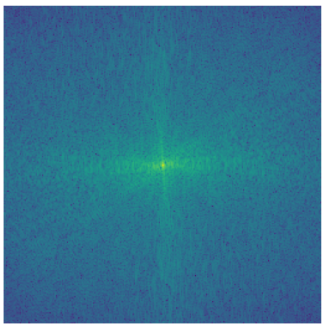
expense of large model sizes, a large number of training images, high training complexity, etc. When dealing with manipulated images, DNN classifiers adopt deeper networks and augment the training set by including all kinds of manipulated images, leading to even larger model sizes and higher training complexity. To address these two problems, we develop a robust and green GAN-fake image detector, named RGGID, in this chapter.

Our RGGID detector is designed based on the assumption that GANs fail to synthesize high-frequency components in local regions, such as edges and textures, in high fidelity. Following [108] and [86], we show real and fake horse images in the pixel- and the spectral-domains in Fig. 3.1. As compared with the real spectral image, the fake spectral image contains artifacts in diagonal and anti-diagonal directions. This corroborates our assumption that GANs do not synthesize high-frequency components well. Here, we focus on complex local regions that have high-frequency components and employ a set of local filters, called filter banks or PixelHops, to extract features. We develop an ensemble scheme to ensure robust detection under different image manipulations. The RGGID solution outperforms DNN-based GAN-fake image detectors in detection performance. Furthermore, it has three additional advantages: 1) low computational and memory complexity (i.e., green), 2) robustness against image manipulations and 3) mathematical transparency.

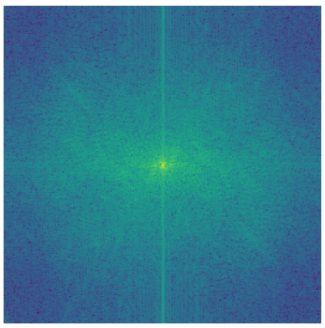
The main contribution of this chapter lies in the study of robustness of RGGID against common image manipulations. Image manipulations introduce additional artifacts to real/fake images. They tend to mask the differences between real/fake images and make the detection problem even more challenging. It is shown by experimental results that RGGID is robust against image manipulations. Other than robustness against image manipulations, the generalization ability of RGGID on unseen dataset is also evaluated. With the rapid development of image generation models, generalization ability is an essential metric for fake-image detectors. A unified fake-image detector against various generative architectures and dataset is preferred by forensics. Experimental result showed that our RGGID method

(a) Real/Pixel-Domain

(b) Fake/Pixel-Domain



(c) Real/Spectral-Domain



(d) Fake/Spectral-Domain

Figure 3.1: Examples of real/fake image pairs (top) and their associated spectral-domain representation pairs (bottom).

can preserve high performance under 11 unseen generative models when trained solely on ProGAN dataset.

The rest of this chapter is organized as follows. The RGGID method is presented in Section 3.2. Experimental results are shown in Section 3.3. The effect of image manipulations on different semantic categories is analyzed and a new experimental setting is presented in Section 3.4. Generalization ability on unseen dataset in evaluated in Section 3.5. This chapter concludes in Section 3.6.

3.2 Proposed RGGID Method

An overview of the proposed RGGID method is given in Fig. 3.2. It consists of the following four modules:

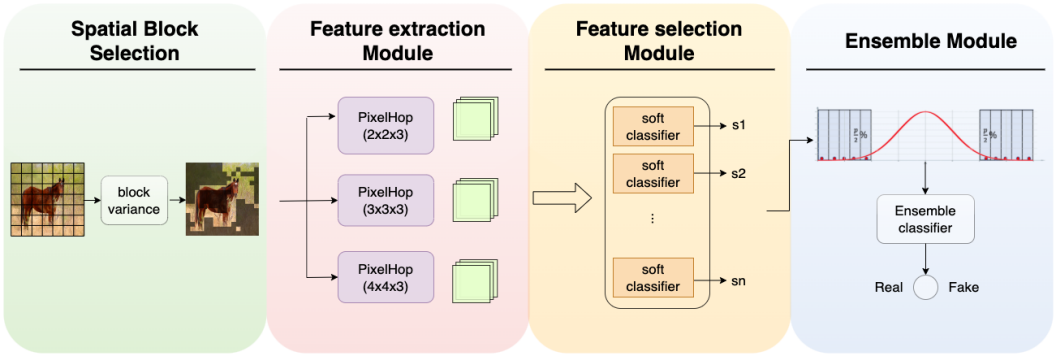


Figure 3.2: An overview of the proposed RGGID method.

1. **Spatial block selection.** We select blocks that contain a substantial amount of high frequency components.

2. **Feature extraction via parallel PixelHops.** We conduct local spectral analysis by studying frequency responses of multiple sets of local filters. Each set of local filters is called a filter bank or a PixelHop.

3. **Discrimant feature selection and block-level decision making.** We use the validation dataset to identify discriminant channels and use their channel responses as features for the block-level soft decisions.

4. **Image-level decision ensemble.** We ensemble the block-level soft decisions to yield the final image-level binary decision.

Each of them will be detailed below.

3.2.1 Spatial Block Selection

Since our method is developed based on the assumption that GAN generators are not able to synthesize high-frequency components in high fidelity, we focus on spatial blocks that contain fine details. In the implementation, we partition images into non-overlapping blocks of size 16x16. Each block will be used as an independent unit for feature extraction,

feature selection, and local decision making in the second and third modules. To select blocks containing fine details, the variance of image pixels in a block is computed. That is, we remove the block mean and sum the squares of pixel residuals. Blocks with a larger variance are selected since they contain more energy of high frequency components. Fig. 3.3 shows examples of selected blocks overlaid with the original images. It is evident that selected blocks are from high-frequency regions, such as the horse head and legs in the horse image, trees in the winter image, cars and buildings in the cityscape image, etc.

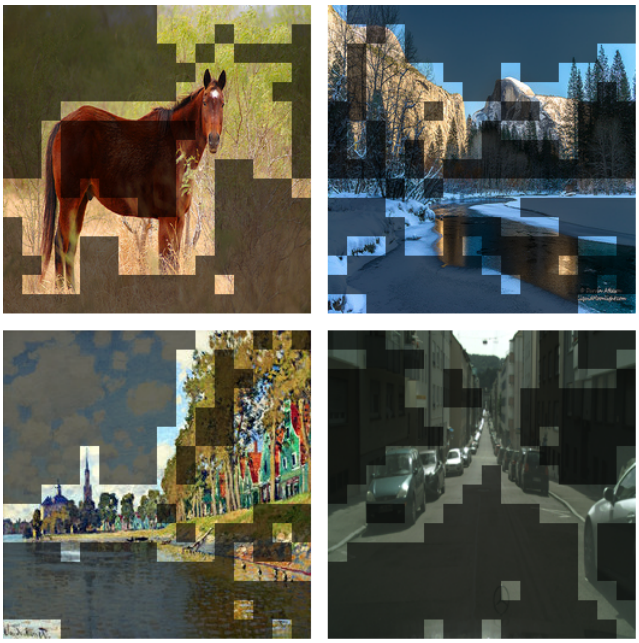
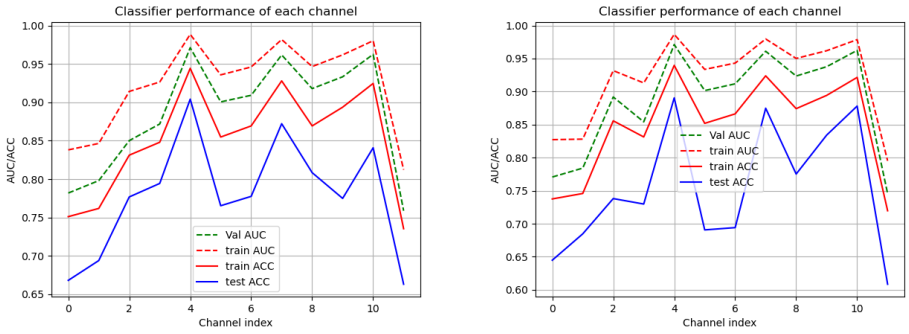


Figure 3.3: Examples of selected spatial blocks from images, where masked blocks are dropped in further analysis.

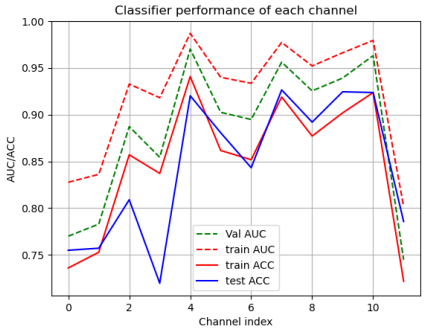
3.2.2 Feature Extraction via Parallel PixelHops

For a squared region of spatial dimension $s \times s$ and spectral dimension c, we can define a local neighborhood of dimension $s \times s \times c$. For example, we can set $s = 2$ and $c = 3$ (due to the R, G, B channels of color images). Then, the neighbood has a dimension of 12 (i.e., 12 pixel values). We can consider different weighted sums of these 12 pixel values, which

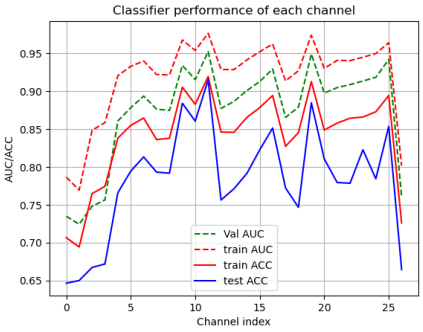


(a) Hor2zeb

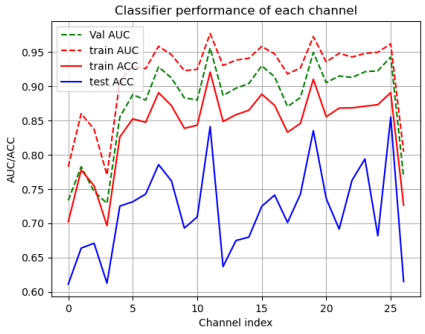
(b) Facade



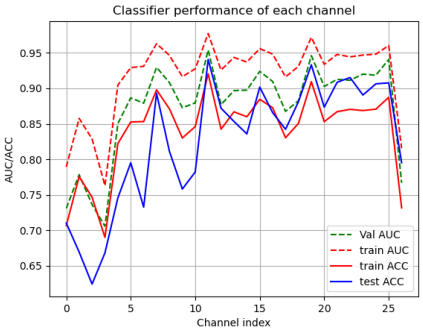
(c) Ukiyoe



(d) Hor2zeb



(e) Facade



(f) Ukiyoe

Figure 3.4: The detection performance on raw images of three exemplary categories

defines a set of filters. The set of filters is called a filter bank. One specific way to define the filter weights is described as follows.

- One DC filter, where all filter weights are set to the same value (i.e., a constant-value vector), and then the vector length is normalized to unity. This filter is called the DC filter and its response is called the DC response.

- Eleven AC filters, where the DC response is subtracted from all pixel values to yield the AC values, principal component analysis is conducted on a collection of neighborhoods, and the eigenvectors associated with non-zero eigenvalues define eleven AC filters.

A filter bank with its filter coefficients selected by this procedure is called a PixelHop. As shown in Fig. 3.2, a PixelHop is used as a feature extraction unit. A PixelHop system is determined by 4 parameters:

1. neighborhood size $s_1 \times s_2$, where s_1 and s_2 denote the width and the height, respectively. Typically, we choose squared neighborhoods such that $s_1 = s_2 = s$;

2. number of spectral components of a pixel, denoted by c;

3. the stride number, denoted by d, which indicates the amount of movement of the neighborhood horizontally or vertically.

In our design, we select three squared neighborhoods of sizes 2×2, 3×3, and 4×4. The spectral component number, c, is equal to 3, and the stride number, d, is one. We apply the three PixelHops to blocks of size 16×16 in parallel without padding. As a result, they have 12, 36 and 48 filter (or channel) responses at 15x15=225, 14x14=196, and 13x13=169 spatial locations, respectively. These responses are call joint spatial-spectral responses. We are interested in channel responses. That is, for a given filter, we collect and order its spatial responses to form a feature vector. For example, for the $2 \times 2 \times 3$ PixelHop, we have 12 channel responses and each of them has a feature vector of dimension 225.

3.2.3 Discriminant Feature Selection and Block-level Decision Making

Different spectral channels have different discriminant power in real/fake image detection. As mentioned earlier, we use the responses at different spatial locations as the feature vector. Furthermore, we adopt a gradient boosting tree algorithm called XGBoost [13] as the classifier. To evaluate the discriminant power of a channel, we compare the classifier

performance on training, validation and test datasets with the area-under-the-curve (AUC) and the accuracy (ACC) metrics. To give an example, we plot the performance curves of each channel for three semantic categories in the CycleGAN dataset [108] with two PixelHops in Fig. 3.4, where the x-axis indicates the channel index. In this example, the images are raw real/fake images without any image manipulations. The x-value ranges from 0 to 11 in (a)-(c), in which a PixelHop of size $2 \times 2 \times 3$ is used. The x-value ranges from 0 to 26 in (d)-(f), in which a PixelHop of size $3 \times 3 \times 3$ is used. Each subfigure shows four performance curves: training AUC (red dashed line), training ACC (red line), validation AUC (green dashed line) and test ACC (blue line). It is evident from the figure that some channels are more discriminant than others. Furthermore, the training, validation and testing datasets share the same discriminant channels. We select those channels with higher validation performance as target channels and train an XGBoost classifier for each channel. In the inference stage, we apply an XGBoost classifier to the spatial responses of the associated channel to obtain a soft decision ranging from 0 to 1, which indicates the probability of the block to be a real or fake image block.

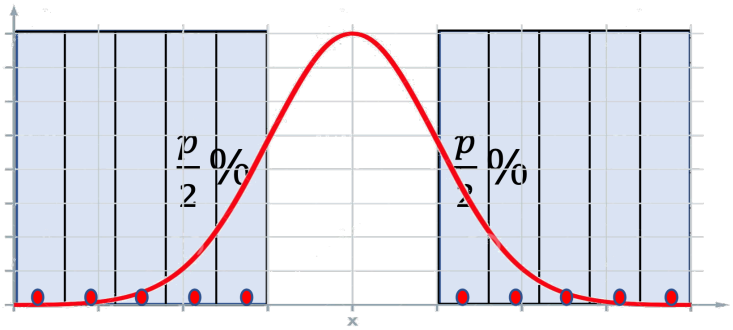


Figure 3.5: Illustration of the block sampling strategy for the image-level decision ensemble.

3.2.4 Image-level Decision Ensemble

Given block-level soft decisions from a single image in the third module, we develop an ensemble scheme to yield the final image-level decision in the last module. We first arrange

the block-level soft decisions from smallest to largest in the unit interval, i.e., [0,1]. The decision scores at the two ends are more informative than those in the middle range. Suppose that we plan to sample $p\%$ of blocks to train an ensemble classifier that will yield the image-level decision. Our sampling strategy is to choose $0.5p\%$ soft decisions from the two ends of the distribution as shown in Fig. 3.5, where selected representative soft decisions are denoted by red dots. The number p is a hyper-parameter that can be decided according to the performance of the validation data.

3.3 Experiments on CycleGAN

3.3.1 CycleGAN Dataset

We evaluated our model on the CycleGAN dataset in [108]. It has 14 semantic categories: Apple, Orange, Horse, Zebra, Yosemite summer, Yosemite winter, Facades, CityScape Photo, Satellite Image, Ukiyo-e, Van Gogh, Cezanne, Monet and Photo. According to the image translation content, the dataset contains 10 subsets where each subset contains both real and translated images. For example, the *hor2zeb* subset includes real horse and zebra images for training CycleGAN and corresponding fake horse and zebra images generated from the trained model. In total, there are over 36k images in the CycleGAN dataset.

We conducted experiments using the Leave-One-Out setting, as was done in [108] and [86]. Namely, one semantic category will be set aside for validation and the remaining semantic categories will be used for training. In this case, our proposed method is not restricted to a specific semantic category and can generalize well to all CycleGAN images. We tested our model under 3 different image manipulation techniques: JPEG compression, image resizing and additive noise. For each type of manipulation, both the training and testing images will be processed with the same manipulation setting to avoid mismatch. First, we discuss the detection on raw image data as reference. Then, we examine the

scenarios in which the various manipulations are applied. We noticed that there exists a few categories that are relatively sensitive to manipulations. In Section 3.4, we analyze the effect of manipulations on sensitive categories and demonstrate that, by including a small amount of images from the sensitive categories in the training stage, our RGGID method is robust to image manipulations for all semantic categories in the CycleGAN dataset.

3.3.2 Detection on Raw Images

Table 3.1 shows the test detection results on the raw CycleGAN dataset with only 10% training data. We compare the proposed RGGID method with six state-of-the-art models. The highest performance we obtained is 99.0% test accuracy acquired from the fusion of 12 channels, in which we select the 4 best channels from each of the three filter banks (i.e., the $2\times2\times3$, $3\times3\times3$, and $4\times4\times3$ filter banks). The second best is 98.8% from 9 channel fusion, in which we select the 3 best channels from each filter bank. The 6 channel fusion result is the same as the one presented in [115]. This is the case where we achieve equally good performance but with the smallest model size. This indicates that our PixelHop solution is very powerful even if only a few channels are selected in the ensemble process.

Table 3.1: Test accuracy on raw images with 10% training data.

Accuracy	ap2or	hor2zeb	win2sum	citysc.	facades	map2sat	Ukiyoe	Van Gogh	Cezanne	Monet	average
DenseNet	79.1	95.8	67.7	93.8	99.0	78.3	99.5	97.7	99.9	89.8	89.2
XceptionNet	95.9	99.2	76.7	100.0	98.6	76.8	100.0	99.9	100.0	95.1	94.5
InceptionNet	85.0	94.8	58.8	99.4	94.0	70.5	99.8	98.8	99.9	89.9	89.1
Cozzolino2017	99.9	99.9	61.2	99.9	97.3	99.6	100.0	99.9	100.0	99.2	95.1
Auto-Spec	98.3	98.4	93.3	100.0	100.0	78.6	99.9	97.5	99.2	99.7	97.2
Nataraj2019	99.7	99.8	99.8	80.6	92.0	97.5	99.6	100.0	99.6	99.2	96.8
RGGID (6 ch)	99.2	99.8	100.0	94.4	100.0	94.1	100.0	100.0	100.0	99.4	98.7
RGGID (9 ch)	99.2	99.7	100.0	94.4	100.0	95.8	100.0	100.0	100.0	99.2	<u>98.8</u>
RGGID (12 ch)	99.2	99.9	100.0	95.9	100.0	95.8	100.0	100.0	100.0	99.1	**99.0**

3.3.3 JPEG Compression Manipulation

To assess the robustness of RGGID under realistic scenarios, we run experiments in which images are compressed using different quality factors. JPEG compression creates distortions such as blocking and ringing artifacts that interfere with the up-sampling artifact originating from generative models. We verified the assumption that high-frequency responses are more distinguishable than other frequencies for the raw data. However, when applying JPEG compression, the high-frequency components of real compressed images are severely distorted as well. As a result, the difference between real and fake images is less discernible. Figure 3.6 shows the soft classification performance for each spectral channel on compressed images with quality factor 85. For subfigures (a)-(c), the filter size is $2 \times 2 \times 3$ (i.e., the number of channels is 12), while for subfigures (d)-(f), the filter size is $3 \times 3 \times 3$ (i.e., the number of channels is 27). For each subfigure, we show five performance curves: train set AUC (red dashed line), train set ACC (red line), validation set AUC (green dashed line), test set AUC (blue dashed line) and test set ACC (blue line). We see that discriminant channels are shifted from high-frequency bands to mid-and-low frequency bands.

We chose three commonly-used quality factors, i.e., 75, 85 and 95, in the experiments. Table 3.2 shows the test accuracy of RGGID for JPEG compressed images. For each quality factor, we show results for individual filter banks as well as for ensemble settings. Results for individual filter banks are marked as $2 \times 2 \times 3$ *only*, $3 \times 3 \times 3$ *only*, and $4 \times 4 \times 3$ *only*. Results for ensemble schemes are marked as *ensemble*. For example, *ensemble (2&3)* means that we use only discriminant channels from the $2 \times 2 \times 3$ and $3 \times 3 \times 3$ filter banks. On the other hand, *ensemble (all)* is the case where we use discriminant channels from all filter banks. For each quality factor, we use **bold** to mark the setting with highest average test accuracy, and underline for the setting with the second highest accuracy. Generally speaking, $2 \times 2 \times 3$ *only* tends to have better performance than other settings. This could

be attributed to the 8×8 block DCT transform used in JPEG. Also, the $2 \times 2 \times 3$ filter bank is more favorable than the $4 \times 4 \times 3$ filter bank. This is because we select the same number of channels from each filter bank. Feature maps of selected channels in the $2 \times 2 \times 3$ filter bank are more informative. This also explains the reason why ensemble schemes do not always give the best result. Also, we see that the accuracies for the *ap2or* and *map2sat* categories are significantly lower than other categories, which will be analyzed in Section 3.4.

Furthermore, we compare RGGID with other state-of-the-art methods in Table 3.3. Here, we present results from 8 other state-of-the-art methods whose performance scores are taken from [68]. In Table 3.3, the first 5 models are relatively shallow networks while the last three (DenseNet, InceptionNet, and XceptionNet) are deeper neural networks. Their performance scores are based on Twitter-like compression as explained in [68]. However, their compression quality factor is not explicitly provided. For fair comparison, we average our best result for each quality factor and present it in the last row of Table 3.3. In terms of the average test accuracy across all semantic categories, RGGID is very close to the two best models, DenseNet and XceptionNet, with only a 0.06% and 0.58% performance gap, respectively.

3.3.4 Image Resizing Manipulation

Another common image manipulation is resizing. We focus on the scenario of resizing to lower spatial resolutions, referred to as down-sizing. Since the down-sizing operation interacts with artifacts arising from up-sampling in generative models and the differences between real and fake images becomes obscure, down-sized fake images are more challenging to detect.

There is little work on detecting resized real/fake images. Zhang *et al.* [108] chose 4 image sizes and randomly selected one as the target size. They trained a neural network

Table 3.2: Test accuracy of RGGID on JPEG compressed images.

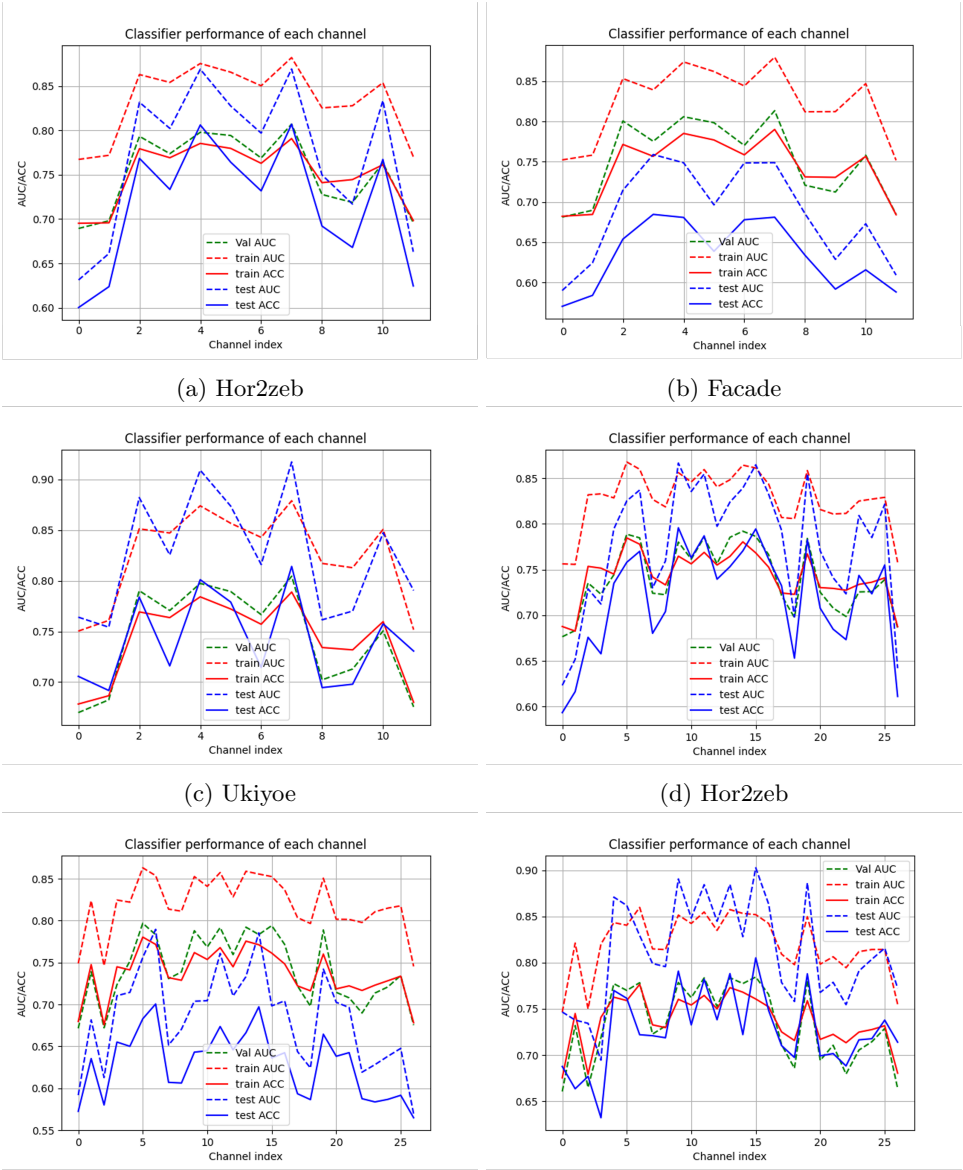
Quality Factor	Setting	ap2or	hor2zeb	win2sum	citysc.	facades	map2sat	Ukiyoe	Van Gogh	Cezanne	Monet	ave.
QF=75	2x2x3 only	67.58	89.21	90.31	93.78	89.50	70.98	90.53	98.83	98.97	81.40	**87.11**
	3x3x3 only	66.11	88.25	87.32	68.74	93.75	69.98	87.80	89.53	98.87	81.96	83.23
	4x4x3 only	63.43	89.09	86.16	89.06	93.38	60.40	83.53	80.47	97.50	81.03	82.40
	ensemble (2&3)	68.05	90.38	90.99	81.24	93.00	60.03	89.17	93.33	99.47	80.93	84.66
	ensemble (2&4)	67.08	88.28	90.58	91.29	93.13	59.35	86.83	93.37	97.70	81.36	84.90
	ensemble (all)	65.91	89.96	88.90	89.58	92.88	63.91	88.43	94.80	98.03	81.08	85.34
QF=85	2x2x3 only	70.66	91.36	90.13	97.19	91.63	50.23	93.07	98.73	99.83	81.22	86.36
	3x3x3 only	62.31	92.17	93.43	97.18	95.75	51.92	94.20	93.83	99.83	82.25	86.29
	4x4x3 only	63.28	92.00	92.80	89.83	93.88	69.66	94.20	91.10	98.70	80.95	86.64
	ensemble (2&3)	72.64	91.69	95.10	87.65	95.63	62.68	96.23	98.33	99.80	82.35	88.21
	ensemble (2&4)	73.58	92.17	94.92	95.90	94.50	60.58	95.03	96.57	99.50	81.82	88.46
	ensemble (all)	72.64	92.46	95.30	95.83	95.75	67.52	96.20	96.53	99.70	82.34	**89.43**
QF=95	2x2x3 only	66.91	92.88	97.65	96.52	95.88	50.00	98.80	99.73	99.97	89.87	**88.82**
	3x3x3 only	64.90	95.46	97.58	91.92	95.13	50.00	99.20	97.37	99.97	89.27	88.08
	4x4x3 only	63.51	95.81	96.90	88.72	96.38	53.47	98.57	92.70	99.53	85.56	87.16
	ensemble (2&3)	67.08	94.29	97.72	92.37	95.88	50.00	99.20	99.17	99.97	91.10	88.68
	ensemble (2&4)	68.42	94.25	97.49	93.85	95.63	51.41	98.93	97.67	99.83	88.92	88.64
	ensemble (all)	64.42	94.25	97.74	91.06	96.25	51.32	99.23	98.87	99.90	89.42	88.26

Table 3.3: Test accuracy comparison of different detectors for JPEG compressed images.

Category	Steganalysis feat.	GAN discr.	Cozzolino 2017	Bayar 2016	Rahmouni 2017	DenseNet	Inception-Net v3	Xception-Net	RGGID
ap2or	79.39	63.29	79.57	54.64	84.96	78.27	78.6	93.52	69.04
hor2zeb	90.02	91.08	89.82	95.34	98.35	93.44	95.23	93.77	91.52
win2sum	56.66	51.9	53.74	50.27	54.30	66.94	64.54	67.07	94.42
citysc.	92.17	53.14	86.81	54.00	57.60	97.83	96.09	95.11	95.38
facades	73.62	88.75	62.88	90.63	91.88	98.19	90.14	99.22	93.71
map2sat	69.39	79.35	89.64	52.69	54.93	80.45	63.84	67.97	62.83
Ukiyoe	65.83	76.56	67.67	58.90	96.83	97.54	99.53	99.66	95.18
Van Gogh	95.30	80.32	98.80	74.27	99.63	98.53	96.31	95.18	98.36
Cezanne	94.73	96.41	99.93	99.77	99.77	99.57	100.00	99.97	99.55
Monet	80.89	81.83	87.33	78.60	89.72	83.95	86.21	84.02	84.54
average	81.09	73.33	82.62	69.17	80.97	88.51	87.37	**89.03**	88.45

with CycleGAN and Auto-GAN horse images, and tested it on other categories. Here, we consider 2 resizing factors (0.5 and 0.75) and conduct experiments under the "Leave-One-Out" setting for all categories.

Table 3.4 shows the results for individual filter banks as well as for ensemble settings. The proposed RGGID method can achieve a maximum accuracy of 95.45% and 92.84% for resize factors of 0.75 and 0.5, respectively. As compared with the 99% detection accuracy



(a) Hor2zeb (b) Facade

(c) Ukiyoe (d) Hor2zeb

(e) Facade (f) Ukiyoe

Figure 3.6: Soft classification performance of exemplary categories on JPEG compressed images (QF=85)

on raw images in Table 3.1, the accuracy degrades by 3.55% and 6.16% for resize factors of 0.75 and 0.5, respectively. Thus, RGGID is robust with respect to image resizing.

For a resize factor of 0.75, the $3 \times 3 \times 3$ filter bank yields the best performance while the ensemble of the $3 \times 3 \times 3$ and $2 \times 2 \times 3$ filter banks yields the second best performance. For

Table 3.4: Test accuracy of RGGID for resized images.

Resize Factor	Setting	ap2or	hor2zeb	win2sum	citysc.	facades	map2sat	Ukiyoe	Van Gogh	Cezanne	Monet	average
0.75	2x2x3 only	97.89	95.67	99.73	64.83	96.75	99.13	99.87	99.47	99.13	95.92	94.84
	3x3x3 only	95.38	95.84	99.84	79.80	93.13	95.03	99.60	98.97	99.60	97.32	**95.45**
	4x4x3 only	98.14	98.40	99.47	53.16	99.63	63.77	99.30	97.50	99.83	97.88	90.71
	ensemble (2&3)	98.39	96.31	99.91	73.38	94.13	95.62	99.80	99.63	99.56	97.10	<u>95.38</u>
	ensemble (all)	98.36	97.06	99.89	73.79	98.88	88.50	99.90	99.33	99.80	97.55	95.31
0.5	2x2x3 only	95.63	95.48	99.31	50.81	93.25	94.71	99.30	94.60	98.73	87.34	90.92
	3x3x3 only	91.88	96.79	97.99	55.92	95.88	96.35	98.70	94.70	94.00	83.48	90.57
	4x4x3 only	94.51	96.08	97.27	75.39	88.50	90.37	98.50	93.00	97.93	85.65	91.72
	ensemble (2&3)	94.69	96.94	99.27	55.98	95.75	98.04	99.17	95.27	98.13	86.64	<u>91.99</u>
	ensemble (all)	96.67	96.33	98.84	70.03	92.75	92.61	99.50	95.30	98.47	87.93	**92.84**

a resize factor of 0.5, individual filter banks are less effective, and the ensemble of all three filter banks gives the best performance.

3.3.5 Additive Gaussian Noise Manipulation

The third image manipulation tested is additive Gaussian noise. Although it may not be as common as JPEG compression and image resizing in social media, Gaussian noise could be used to cover up certain weaknesses in synthesized images. It is essential to demonstrate the robustness of RGGID against additive Gaussian noise. In our experiments, we normalize the pixel values of the raw image to $[0, 1]$ and use Gaussian noise with two noise levels (namely, $\sigma = 0.01$ and 0.02) to simulate a realistic scenario in forensics. Because additive noise introduces additional high-frequency information to the raw image, the source differences between real and fake images in high-frequency regions are diminished. This phenomenon is observed in the soft classification performance shown in Figure 3.7, where $\sigma = 0.01$. In Figure 3.7, (a)-(c) uses the $2 \times 2 \times 3$ filter bank so that the x-axis has 12 channels, (d)-(f) uses the $3 \times 3 \times 3$ filter bank so that the x-axis has 27 channels. Each subfigure shows four performance curves: train set AUC (red dashed line), train set ACC (red line), validation set AUC (green dashed line) and test set ACC (blue line). Similar to Figure 3.6, we see that high-frequency channels are not as discriminant as those in the raw image dataset. Discriminant channels are shifted from high-frequency to mid-frequency bands.

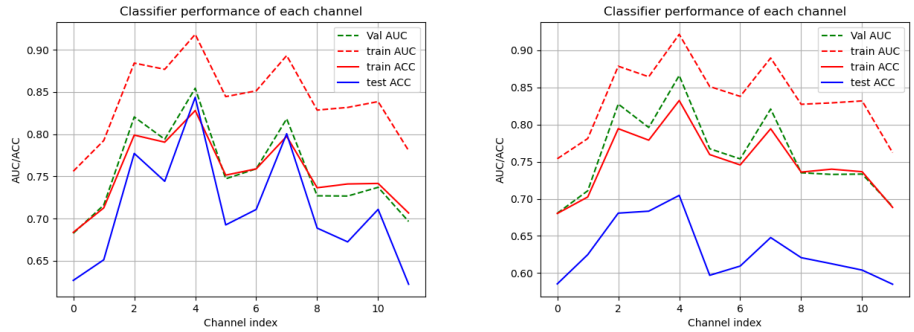
Table 3.5 shows the test accuracy for each semantic category under different noise levels. When $\sigma = 0.01$, RGGID can achieve a maximum average accuracy of 89.89%, which is approximately a 10% drop as compared to the accuracy for the raw image dataset. When the noise level is increased to $\sigma = 0.02$, noise in the smooth regions is visible to human eyes, and detection of fake images becomes more challenging. In this case, the maximum average accuracy of RGGID is 86.52% using the $3 \times 3 \times 3$ filter bank. Overall, RGGID can maintain good detection performance against additive Gaussian noise.

Table 3.5: Test accuracy of RGGID for noisy images

σ	Setting	ap2or	hor2zeb	win2sum	citysc.	facades	map2sat	Ukiyoe	Van Gogh	Cezanne	Monet	average
$\sigma = 0.01$	2x2x3 only	64.42	91.71	95.96	92.52	95.5	52.24	93.07	99.63	99.83	99.49	88.44
	3x3x3 only	65.27	94.81	99.13	75.29	96.25	52.11	99.27	99.97	99.93	99.88	88.19
	4x4x3 only	59.51	96.13	98.70	53.31	95.62	72.39	98.77	99.90	99.23	98.64	87.22
	ensemble (2&3)	64.70	94.69	94.12	93.28	96.38	51.93	97.33	99.63	99.87	99.81	89.17
	ensemble (3&4)	64.50	93.73	98.77	92.84	96.38	53.81	99.17	99.83	99.97	99.88	**89.89**
	ensemble (all)	62.39	94.69	94.12	93.28	96.38	53.63	97.73	99.90	99.90	99.81	<u>89.18</u>
$\sigma = 0.02$	2x2x3 only	68.20	90.52	94.80	51.65	90.00	63.96	97.67	98.80	98.90	96.65	85.12
	3x3x3 only	66.08	89.53	96.90	64.84	91.13	65.92	97.53	97.83	98.93	96.58	**86.52**
	4x4x3 only	69.76	93.17	94.19	51.83	89.13	76.94	95.70	97.83	97.20	97.86	<u>86.36</u>
	ensemble (2&3)	67.85	91.84	97.10	53.31	91.13	65.10	97.63	98.07	98.47	96.62	85.71
	ensemble (3&4)	65.47	92.50	93.14	64.17	87.50	67.35	96.63	98.87	99.47	98.17	86.32
	ensemble (all)	64.23	92.61	94.12	53.34	86.38	63.61	96.20	98.40	99.27	98.17	84.63

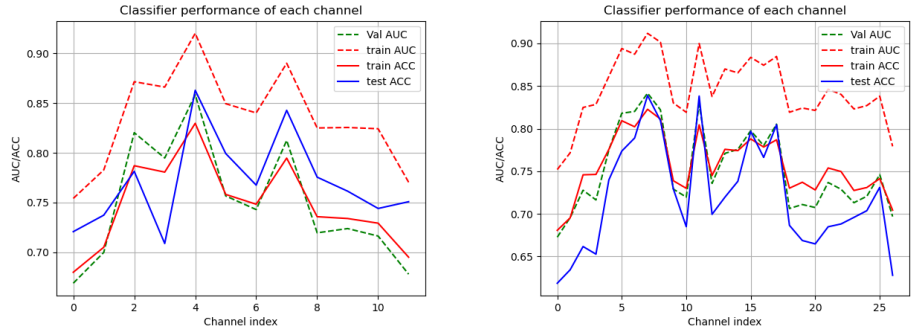
3.4 Analysis

For each of the three aforementioned image manipulations, there are certain categories for which the performance is significantly lower as compared to the remaining categories. They are referred to as challenging categories. They are *ap2or* and *map2sat* for JPEG compression, *citysc* for image resizing, and *ap2or*, *citysc* and *map2sat* for additive Gaussian noise. We first analyze the effect of image manipulations in Section 3.4.1. Next, we propose a new experimental setting called *Leave-None-Out* in Section 3.4.2. We conduct extensive experiments under the new setting and show that the performance for the challenging categories can be increased significantly.



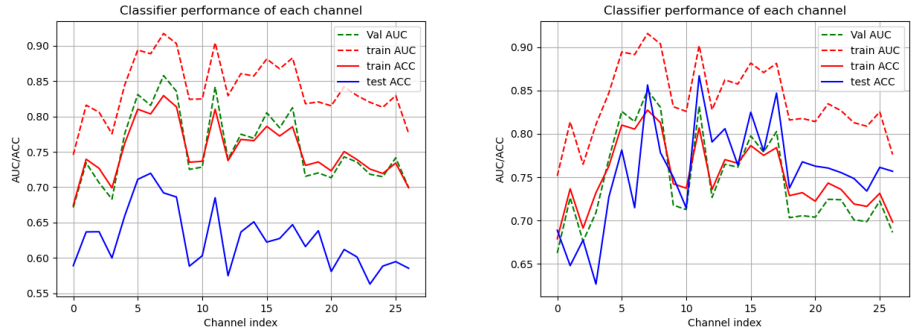
(a) Hor2zeb

(b) Facade



(c) Ukiyoe

(d) Hor2zeb



(e) Facade

(f) Ukiyoe

Figure 3.7: Soft classification performance of six exemplary semantic categories on noisy images ($\sigma = 0.01$)

3.4.1 Image Manipulation Analysis

For JPEG compression, the two challenging categories are *ap2or* and *map2sat*. In figure 3.8, we show exemplary images from *hor2zeb*, *Ukiyoe*, *ap2or* and *map2sat* categories (in the 1^{st} column), JPEG compressed images (in the 2^{nd} column), difference maps between

original and JPEG compressed images (in the 3^{rd} column), spectra of original images (in the 4^{th} column) and spectra of compressed images (in the 5^{th} column). As revealed by the difference maps, we observe stronger distortion on the *ap2or* and *map2sat* images caused by JPEG compression as compared with *hor2zeb* and *Ukiyoe* images, which are considered easy categories. Furthermore, stronger high-frequency components of original images in the *ap2or* and *map2sat* categories are also revealed by their spectra (see the fourth column). On one hand, these high-frequency components cannot be synthesized well in GAN-fake images. On the other hand, they are degraded by JPEG compression for both real and fake images as well. JPEG compression offers a masking effect on the generation artifact in fake image detection with respect to these two challenging categories.

Similarly, we conduct frequency analysis of image resizing in Figure 3.9. We show exemplary images from *citysc*, *Monet* and *facades* categories (in the 1^{st} column), resized images (in the 2^{nd} column), difference maps (in the 3^{rd} column), spectra of original images (in the 4^{th} column) and spectra of resized images (in the 5^{th} column). For ease of comparison, we resize original images to smaller size and resize them back to original size and compute the pixel-wise difference between the two to yield the difference map. Images from the *citysc* category contain street views from car cameras and its content contains many vertical edges. As shown in the figure, resizing introduces stronger vertical distortion on images from the *citysc* category as compared to other categories. These vertical edges in raw images offer good cues for fake image detection. Since these cues are masked by resizing, it becomes more challenging to differentiate real and fake images.

The same phenomenon is observed for the *ap2or*, *citysc* and *map2sat* categories under the additive noise manipulation as shown in Figure 3.10. We show exemplary images from *ap2or*, *map2sat*, *citysc*, *Cezanne* and *win2sum* categories (in the 1^{st} column), noisy images (in the 2^{nd} column), spectra of original images (in the 3^{rd} column), and spectra of noisy images (in the 4^{th} column). By comparing the spectra before and after additive noise, we

see that *ap2or*, *citysc* and *map2sat* images are more affected by additive noise than *Cezanne* and *win2sum* images. It is worthwhile to point out that the difference in spectral image for the satellite map category is not as obvious as that for the *citysc* category. This is because the satellite map images are much larger in size and the scales of their spectral images are actually different.

3.4.2 Leave-None-Out Setting

From analysis in Section 3.4.1, we see that leaving a specific semantic category out during training can affect the performance for certain semantic categories under image manipulation scenarios. For example, if we leave the *ap2or* semantic category out in JPEG compression, its performance becomes much worse as shown in Table 3.2.

Actually, the Leave-One-Out setting is not practical in real-world forensics. It is reasonable to assume that we can have access to all semantic category images when we are confronted with fake image attacks. For this reason, we propose another experimental setting called **Leave-None-Out**, where all semantic categories in the CycleGAN dataset are employed in the training process. In this setting, we enlarge the training dataset by including 10% of test category images and use the other 90% of test category images in testing.

Since this setting only includes a small number of test category images in the training set, we can still examine the detection performance and robustness of our model with respect to a specific semantic category. We conduct experiments under the Leave-None-Out setting for each manipulation and show the results in Tables 3.6, 3.7 and 3.8 for JPEG compression, resizing, and additive noise, respectively. We use * to denote the result under the Leave-None-Out setting. As compared with the Leave-One-Out setting, we observe a 3-20% test accuracy increase for the new setting. For example, for image resizing with a resize factor of 0.5, the test accuracy of RGGID improves from 70.03% to 93.09% for the *citysc* category with an ensemble of all three filter banks.

Figure 3.8: Frequency analysis on JPEG compression (QF=75).

.

3.4.3 Model Size, Computational Complexity and Weak Supervision

The proposed RGGID method is a green solution since it has low computational and memory complexity and it can achieve high performance with weak supervision as discussed below.

Model Size Comparison. We compute the model size for each component in Table 3.9. The model parameters of RGGID include PixelHop filter parameters, soft classifier parameters, and ensemble classifier parameters. The number of soft classifier parameters is proportional to the number of selected channels. For example, for individual filter banks, if the filter size is $s \times s \times c$ and the selected channel number is k, the number of PixelHop filter parameters is $s^2 \times c \times k$. For ensemble schemes, to obtain the total number of PixelHop filter parameters, we sum across the filter banks in the ensemble. For the soft classifier

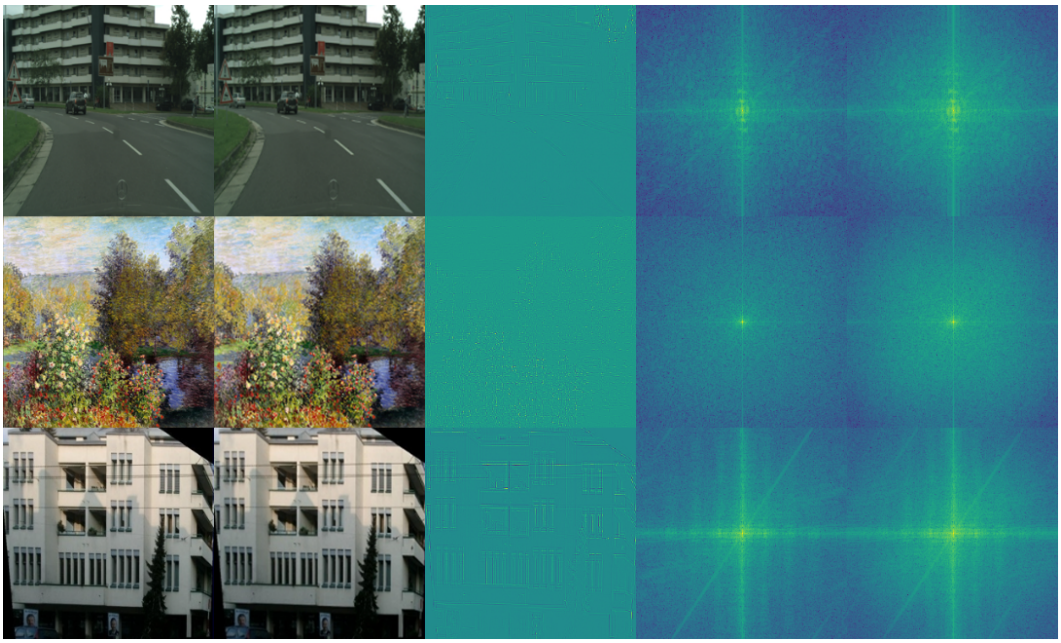


Figure 3.9: Frequency analysis on image resizing (Resize factor=0.75).

Table 3.6: Test accuracy for JPEG-compressed images under the Leave-None-Out setting

	Setting	ap2or	map2sat	ap2or*	map2sat*
QF=75	2x2x3 only	67.58	70.98	72.30	79.21
	3x3x3 only	66.11	69.98	68.74	66.18
	4x4x3 only	63.43	60.40	66.36	74.85
	ensemble (2&3)	68.05	60.03	72.19	69.62
	ensemble (2&4)	67.08	59.35	72.27	74.29
	ensemble (all)	65.91	63.91	72.16	74.85
QF=85	2x2x3 only	70.66	50.23	77.21	71.55
	3x3x3 only	62.31	51.92	72.46	73.63
	4x4x3 only	63.28	69.66	70.25	77.43
	ensemble (2&3)	72.64	62.68	76.35	72.16
	ensemble (2&4)	73.58	60.58	76.88	70.59
	ensemble (all)	72.64	67.52	76.38	70.59
QF=95	2x2x3 only	66.91	50.00	87.09	69.23
	3x3x3 only	64.90	50.00	84.08	75.11
	4x4x3 only	63.51	53.47	81.46	66.23
	ensemble (2&3)	67.08	50.00	86.78	76.19
	ensemble (2&4)	68.42	51.41	86.59	78.26
	ensemble (all)	64.42	51.32	86.70	77.01

parameters, we train each XGBoost classifier with 100 trees, and each tree has a maximum depth of 6. The total number of soft classifier parameters is equal to the number of channels multiplied by 19k. The ensemble classifier is a shallow XGBoost classifier with only 10 trees

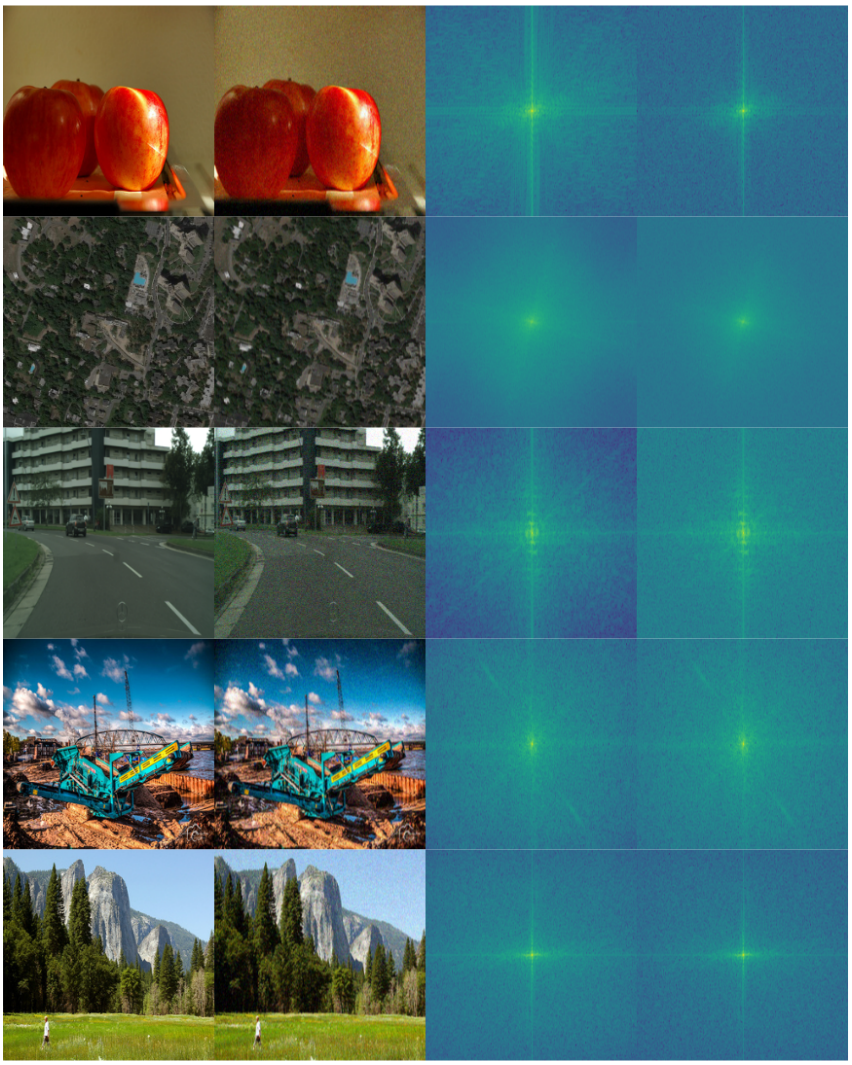


Figure 3.10: Frequency analysis on additive Gaussian noise with $\sigma = 0.02$.

and each tree has a depth of 1. Thus, the number of ensemble classifier parameters is 40. For the various settings, we see that the model size ranges from 76.2k to 231.2k parameters.

The model sizes of other state-of-the-art fake image detection models on the raw CycleGAN dataset are given in Table 3.10. DNN models such as DenseNet, InceptionNet and XceptionNet have millions of parameters. A shallow CNN that has two convolutional layers and one fully connected layer was introduced by Cozzolino *et al.* [21]. Its model has only 1k

Table 3.7: Test accuracy for resized images under the Leave-None-Out setting

	Setting	citysc.	citysc.*
resize factor 0.75	2x2x3 only	64.83	97.48
	3x3x3 only	79.80	99.05
	4x4x3 only	53.16	98.58
	ensemble (2&3)	73.38	99.38
	ensemble (all)	73.79	99.85
resize factor 0.5	2x2x3 only	50.81	82.85
	3x3x3 only	55.92	67.59
	4x4x3 only	75.39	91.24
	ensemble (2&3)	55.98	83.69
	ensemble (all)	70.03	93.09

Table 3.8: Test accuracy for noisy images under the Leave-None-Out setting

	Setting	ap2or	citysc.	map2sat	ap2or*	citysc.*	map2sat*
$\sigma = 0.01$	2x2x3 only	64.42	92.52	52.24	73.58	81.18	88.63
	3x3x3 only	65.27	75.29	52.11	78.90	84.67	81.55
	4x4x3 only	59.51	53.31	72.39	69.37	96.39	90.77
	ensemble (2&3)	64.70	93.28	51.93	75.56	81.18	83.71
	ensemble (3&4)	64.50	92.84	53.81	76.39	84.67	79.33
	ensemble (all)	62.39	93.28	53.63	72.07	81.18	83.71
$\sigma = 0.02$	2x2x3 only	68.20	51.65	63.96	72.76	85.15	72.47
	3x3x3 only	66.08	64.84	65.92	77.30	78.47	81.83
	4x4x3 only	69.76	51.83	76.94	73.84	94.75	86.42
	ensemble (2&3)	67.85	53.31	65.10	76.59	73.63	76.01
	ensemble (3&4)	65.47	64.17	67.35	77.21	78.47	88.42
	ensemble (all)	64.23	53.34	63.61	79.91	73.63	83.14

parameters. Auto-Spec [108] uses ResNet-34 as a classification network and has 21.8M parameters. Nataraj *et al.* [71] used a neural network for feature extraction and classification, and its model size is 730k. In contrast, RGGID has a minimum of 76.2k parameters (with the 2x2x3 filter bank) and a maximum of 231.2k parameters (with the ensemble of all three filter banks). Its model size is significantly smaller than those of DNNs.

Computational Complexity. We measure the training time from scratch on CPU Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz. The average training time for each category is 1.9 hours, yielding a total training time of 19 hours for all 10 categories. Other existing models need GPU and they often rely on pre-trained models.

Table 3.9: Model size breakdown

Ensemble scheme	2x2x3 only	3x3x3 only	4x4x3 only	ensemble (2&3)	ensemble (3&4)	ensemble (2&4)	ensemble (all)
No. of PixelHop filter parameter	12×12	27×27	48×48	873	3033	2448	3177
No. of selected channels	4	4	4	8	8	8	12
No. of soft classifier parameters	76k	76k	76k	152k	152k	152k	228k
No. of ensemble classifier parameters	40	40	40	40	40	40	40
Total No. parameters	76.2k	76.8k	78.3k	152.9k	155k	154.4k	231.2k

Table 3.10: Model size comparison

Method	Number of parameters
DenseNet	1.0M
XceptionNet	22.9M
InceptionNet	23.8M
Cozzolino	1k
Auto-Spec	21.8M
Nataraj	730k
RGGID (raw dataset)	76.2k

Weak Supervision. As reported in Table 3.1, RGGID can achieve an accuracy of 99.0% on the raw image dataset based on 10% of training images from each training category. We also conducted experiments using only 1%, 2%, $\cdots$, 8% and 9% of training data from each semantic category and show the corresponding test accuracies in Figure 3.11 for the $2 \times 2 \times 3$ filter bank. Its test accuracy reaches 93% even with 1% of the original training images. It converges to 99% using only 5% of the original training images. This shows that RGGID can perform well even under extremely weak supervision.

3.5 Generalization on Unseen Dataset

In this section, we evaluate the generalization ability of RGGID method towards unseen generative architecture and datasets. The experimental setup is adopted from [86], which aims

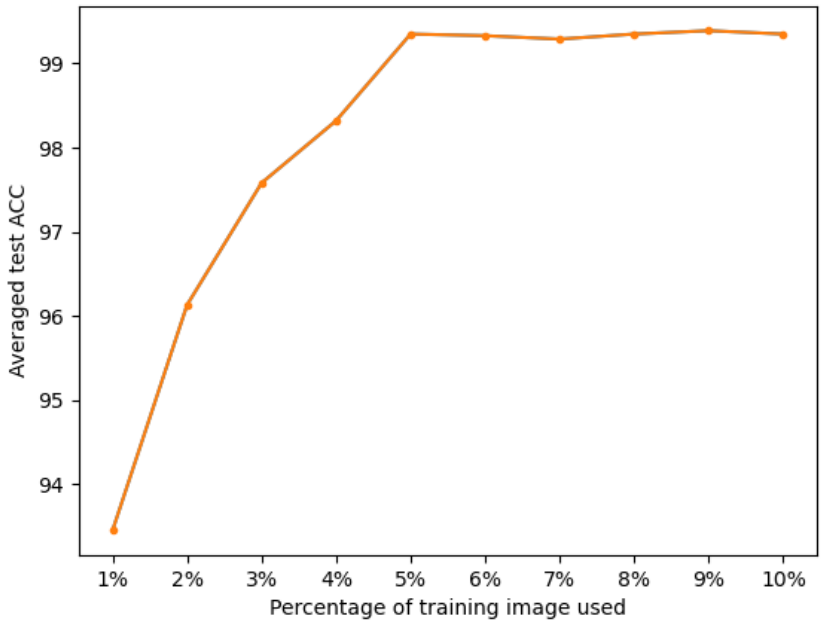


Figure 3.11: The test accuracy as a function of different percentages of the total training images.

to testify CNN-generated images are surprisingly easy to spot as long as the model is trained from intensive augmented data. Same as [86], the dataset we use contains images synthesized by a wide variety of generative models. All of them have an upsampling-convolutional structure. In the training set, fake images from 20 object categories are generated by the Pro-GAN model only. There are 720K real/fake image pairs in the training set and 4K images in the validation set. In the testing set, fake images are generated by the following eleven models: Pro-GAN [42], Style-GAN [43], Big-GAN [8], Cycle-GAN [37], Star-GAN [18], Gau-GAN [73], CRN [12], IMLE [57], SITD [9], SAN [22], and Deepfake [76]. Exemplar images from the 11 generative models are exhibited in Figure 3.12 and Figure 3.13. Among the 11 generative models, CRN [12] and IMLE [57] have much larger image size than others. Their synthesized image can be as large as 3000×4000. While the other generative models, especially GAN-based models have normal image size, such as 256×256. The variation

of semantic categories as well as image size across all 11 generative models make it a good source to evaluate our generalization ability and robust on unseen architectures and dataset.

Table 3.11: Comparison of the average precision of fake-image detectors against eleven generative models in Experiment II. **Boldface** is used to indicate best performance.

Methods	Auto-Spec	Wang *et al.*	Ours (10%)	Ours (20%)	Ours (30%)
Pro-GAN	75.6	100.0	99.9	99.9	99.9
Style-GAN	68.6	96.3	99.9	99.9	99.9
Big-GAN	84.9	72.2	77.1	75.2	74.4
Cycle-GAN	100.0	84	97.8	97.3	96.7
Star-GAN	100.0	100.0	100.0	100.0	99.9
Gau-GAN	61.0	67.0	94.6	94.7	94.8
CRN	80.8	93.5	76.4	86.8	91.3
IMLE	75.3	90.3	92.8	95.3	98.2
SITD	89.9	96.2	76.5	76.5	76.7
SAN	66.1	93.6	84.4	83.8	80.3
Deep-fake	39.0	98.2	94.5	93.3	91.3
mAP	76.5	90.1	90.4	91.2	**91.2**

In terms of experimental details, we follow the procedure specified in [86], by first training RGGID with real and ProGAN-generated fake images, and then evaluating its detection performance on real images or fake images generated by the aforementioned 11 generative models. The performance comparison between RGGID and two existing methods, Auto-Spec and the method proposed by Wang *et al.* in [86], is shown in Table 3.11. It is worthwhile to emphasize three points. First, since we do not include augmentation in the training of our method, we compare against [86] under the no augmentation setting. Second, we evaluate the performance in terms of average precision (AP) so as to be consistent with [86]. Third, we collect a total of 10%, 20% and 30% of samples from the two ends for the image-level ensemble and show the corresponding mean AP (mAP) values. We see from the table that RGGID outperforms both Auto-Spec and the method from [86] in all three cases (i.e., 10%, 20% and 30% of samples) in terms of mAP. RGGID outperforms both Auto-Spec and the method from [86] by a large margin in the case of Gau-GAN. RGGID performs worse in the

Figure 3.12: Exemplar real and synthesized images from 11 generative models (1st part)

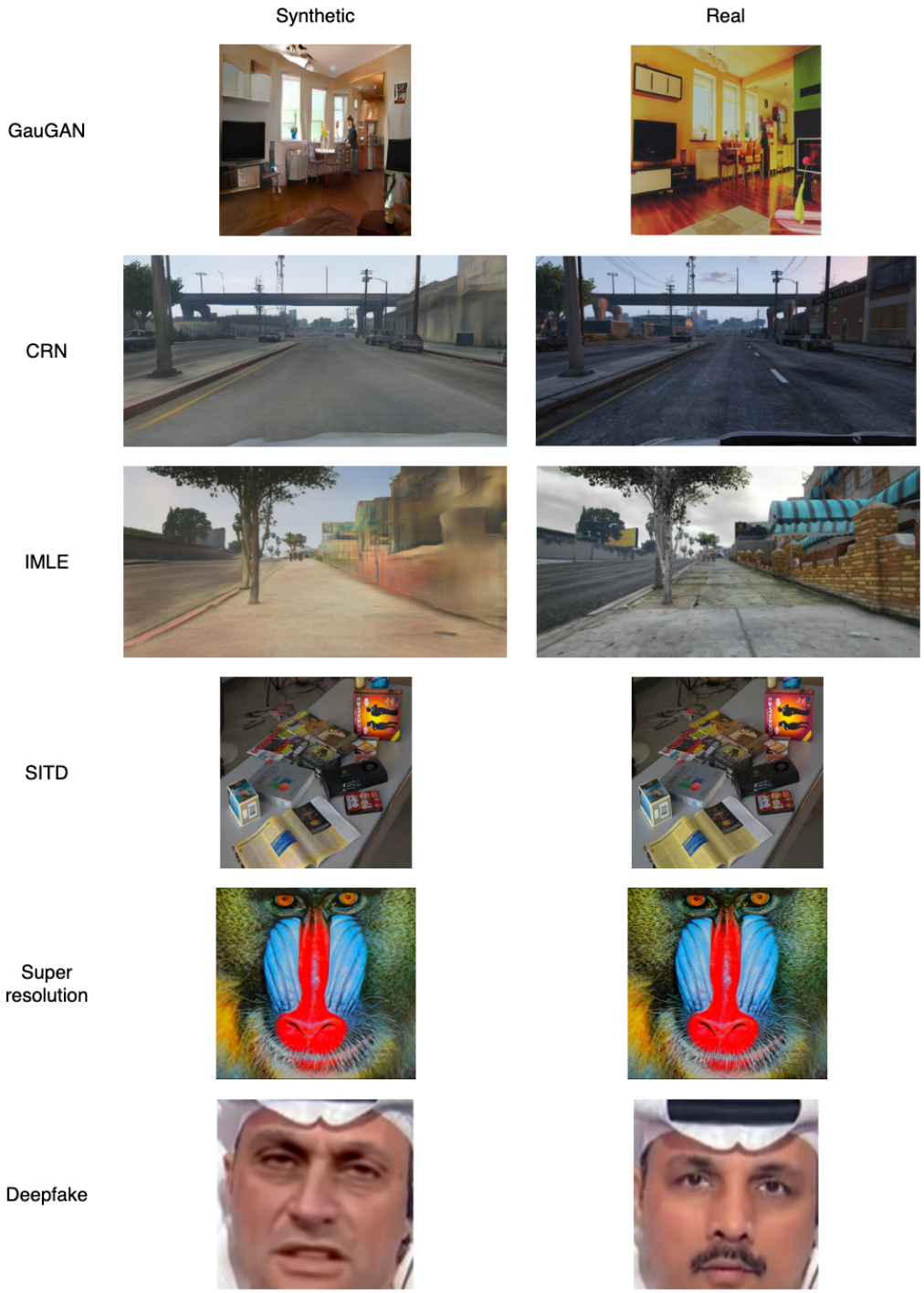


Figure 3.13: Exemplar real and synthesized images from 11 generative models (2nd part)

case of Big-GAN, SITD and SAN, indicating a weaker transferability from ProGAN to these three generative models. We suspect that high resolution images from those two generative models contain less discernible high-frequency components in each selected block. SITD and SAN generate images does not match well with that of the training images generated by ProGAN, which are of size 256×256. One possible fix is to rescale these large images to smaller ones in the pre-processing step.

Based on the experimental results in Table 3.11, we can safely conclude that our RGGID method is robust against unseen generative architecture and datasets. RGGID method outperforms Wang *et al.* method by 1.1% in terms of mAP and Auto-Spec method by a large margin. RGGID does not need intensive data augmentation to preserve high-performance. Thus, it makes our method green and robust.

3.6 Conclusion

A green and robust CNN-generated image detector called RGGID was proposed in this chapter. It was developed under the assumption that GAN-based generative models often fail to generate high-frequency components of real images in high fidelity. Thus, it focuses on complex local regions that have high-frequency components and employs a set of local filters, called filter banks or PixelHops, to extract features. Discriminant channels were identified and their responses were used as features and fed into the XGBoost classifier for soft decision. Finally, various ensemble schemes were adopted to make RGGID adaptive to different semantic categories and robust with respect to compression, resize and additive noise manipulations.

RGGID was evaluated on the CycleGAN and other 11 generative models in this chapter. Under CycleGAN dataset experiment, we successively proved our robustness against

different manipulations such as compression, resize and additive noise. Under 11 generative models experiment, we verified RGGID's generalization ability on unseen generative architectures and dataset. Both experiments validates the light-weight (green), robust and high-performance of RGGID method. In the future, it's interesting to further explore the feasibility of our method under high-resolution scenarios where our assumption of synthetic images lack high-frequency components may not satisfy.

Chapter 4

Green Steganalyzer: A Green Learning Approach to Image Steganalysis

4.1 Introduction

Image steganography aims to embed hidden information in certain source images called cover images. Secret information is concealed in either the pixel domain by slightly modifying the pixel value or by tweaking the DCT coefficients in JPEG-coded images, resulting in stego images. The most secure steganographic technique today is content-adaptive steganography. With this approach, modification of pixels is likely to happen in complex regions such as edges and texture regions, thus making it more difficult to differentiate cover and stego images.

Image steganalysis is a technique to differentiate stego and cover images. The development of more secure steganography algorithms has led to the need for more powerful steganalysis methods. Before the advent of deep learning (DL), steganalysis was developed by using a feed-forward design. Traditional steganalysis usually consist of three major steps: 1) noise residual computation, 2) feature extraction, and 3) binary classification. In the first step, a variety of heuristic high-pass filters are used for noise residual computation, aiming to suppress image content. In the second step, based on noise residuals, higher order

statistics are analyzed and selected as features. In the third step, a machine-learning-based classifier (or an ensemble of several of them) is trained to make final decision.

One famous traditional steganalysis method operating in the spatial domain is the Spatial Rich Model (SRM) [30]. It builds a rich model by taking the union of diverse submodels learned from quantized patches of noise residuals. SRM was shown to be effective against several simpler steganographic schemes under different payloads. However, SRM does not perform well against more secure steganography schemes such as content-adaptive steganography. Furthermore, SRM has an extremely large feature number, leading to a high computational cost.

Due to the great success of DL in computer vision, DL-based steganalysis has received attention since 2015, e.g., Qian-Net [74], Xu-Net [91], Ye-Net [94], Yedroudj-Net [96], Zhu-Net [107], DRN [89], and GBRAS-Net [75]. Although low-, mid-, and high-frequency components all play a role in computer vision tasks such as object classification and detection, the high-frequency information is more relevant to the detection of weak stego-signals in image steganalysis. Thus, DL-based methods follow the first step in traditional steganalysis and use a high-pass filter (HPF) for image preprocessing. HPF can be viewed as a fixed-parameter layer. It converts raw images into noise residual images as the desired input to deep neural networks (DNNs). Through a careful architectural design, the model parameters of a DNN can be automatically determined by the backpropagation algorithm. End-to-end optimized DNNs yield better detection performance than traditional steganalysis methods. However, their underlying decision mechanism is mathematically obsecure. Furthermore, they have a large number of trainable parameters (i.e., a large model size) and demand higher computational complexity in both training and inference.

The shortcomings of traditional and DL-based steganalysis methods motivate this work. Here, we propose an energy-efficient and mathematically transparent steganalysis method and call it the Green Steganalyzer (GS). GS adopts a modular design based on the green

learning paradigm [14,46,48–50]. It is worthwhile to emphasize that we abandon end-to-end optimization as done in DL-based methods but go with the modular design for the purpose of interpretability.

GS consists of three modules: 1) pixel-based anomaly prediction, 2) embedding location detection, and 3) decision fusion for image-level classification. In the first module, GS decomposes an image into patches, adopts Saab transforms [50] for feature extraction from patches, and conducts self-supervised learning to predict an anomaly score of the patch center. In the second module, GS analyzes the anomaly scores of a pixel and its neighborhood to find pixels of higher embedding likelihood. In the third module, GS focuses on pixels of higher embedding probabilities and fuses their abnomaly scores to make final image-level decision.

Compared with traditional and DL-based steganalysis methods, GS has the following three major advantages.

1. High Detection Performance

   We compare the detection error rates of various steganalysis methods against S-UNIWARD, WOW and HILL steganography schemes under two payloads in Sec. 4.4.2. It is observed that GS outperforms SRM (a traditional non-DL-based steganalyzer) by a significant margin. GS also outperforms many DL-based steganalyzers. Its performance is comparable with two state-of-the-art DL-based steganalyzers (i.e. Zhu-Net and GBRAS-Net).

2. Small Model Size and Low Complexity

   We compare the model sizes and the floating point operation numbers (FLOPs) per pixel in the inference stage of DL-based steganalyzers and GS in Sec. 4.4.3. It is shown that GS has a substantially smaller mode size and a much lower computational complexity.

3. Mathematical Transparency

   As compared with DL-based steganalyzers, GS is mathematically transparent due to its modular design. One can gain a clear understanding of each individual module.

The rest of the paper is organized as follows. We give a high-level overview on three image steganalysis approaches; namely, traditional, DL-based, and the proposed green-learning-based (GL-based) steganalyers in Section 4.2. Then, we elaborate on the proposed GS in Section 4.3. Experimental results on the detection performance and the model sizes and complexity analysis are presented in Section 4.4. Finally, concluding remarks and future extensions are given in Section 4.5.

4.2 Review of Related Work

Related previous work is reviewed in this section. First, we conduct a brief survey on traditional and DL-based image steganalyzers in Sections 4.2.1 and 4.2.2, respectively. Then, we present the emerging green learning paradigm in Section 4.2.3.

4.2.1 Traditional Image Steganalysis

Traditional steganalysis methods use hand-crafted filters to extract a set of basic features from images. Afterward, statistical tools, such as the histogram or the co-occurrence matrix, are used to derive more advanced features. Finally, image features are fed into a machine learning classifier for the final decision of stego or cover images.

Fridrich *et al.* [30] proposed the Spatial Rich Model (SRM) by considering diverse relationships between pixels. Rich submodels were constructed from the noise component of images. Features from all submodels were concatenated to form a high-dimensional feature vector and fed into ensemble support vector machine (SVM) classifiers to yield the

final decision. SRM has proven to be successful against a wide spectrum of steganographic schemes.

Lu *et al.* [63] used the Fisher criterion to reduce the dimension of steganalytic feature vector. They analyzed the separability of single-dimension and multiple-dimension spatial-domain features by using the Euclidean distance. Furthermore, they used the Fisher criterion to select feature components with best separability as the final steganalytic features. Tang *et al.* [83] assigned different weights to pixels based on their embedding probabilities in the feature extraction process. That is, pixels with higher embedding probabilities were assigned larger weights. Their scheme was effective especially under a low embedding rate (i.e., lower than 0.20 bpp) among traditional image steganalysis methods.

4.2.2 DL-based Image Steganalysis

4.2.2.1 Earlier Work

State-of-the-art image steganalysis methods are dominated by DL-based solutions. Qian *et al.* [74] proposed one of early neural network models, called GNCNN, for image steganalysis. It contained a learnable convolutional neural network (CNN) model with a fixed high-pass filter as its preprocessing layer. It utilized the Gaussian function as non-linear activation in the convolutional layers. As compared with classical steganalyzers, GNCNN was the first one to automate the feature-extraction and classification steps in a unified system. It achieved performance comparable with that of classical methods against HUGO, S-UNIWARD and WOW three steganographic schemes under various payloads from 0.3 bpp to 0.5 bpp. Xu *et al.* [91] also used a fixed high-pass filter layer to obtain noise residuals as the input to learnable neural networks. Then, they computed the absolute values of features from 5 groups of convolutional modules and used $Tanh$ as the activation function. Ye *et al.* [94] proposed a solution called the Ye-Net. It utilized the filter banks from SRM as the

initialization weights of the first convolutional module. Furthermore, it incorporated the selection-channel aware (SCA) knowledge in the design of the CNN architecture and adopted a novel activation, called the Truncated Linear Unit (TLU), to better suit the nature of stego-noise (± 1). Yedroudj *et al.* [96] proposed another CNN-based model called Yedroudj-NET. It brought together the merits of its predecessors; e.g., the use of predefined high-pass filter banks from SRM as a preprocessing step and the adoption of both absolute Value (ABS) and TLU activations in the convolutional module.

4.2.2.2 Recent Work

DNNs have become popular in recent years due to their better performance. Several advanced DL-based image steganalyzers have been proposed based on DNNs to achieve higher detection perofrmance at the expense of a higher computational cost.

Wu *et al.* [89] utilized the ResNet architecture for steganalysis, which they called DRN (deep-residual-learning-based network). Again, a high-pass filter was used as a preprocessing step. The residual learning blocks was adopted to preserve features from weak stego signals. Boroumand *et al.* [7] proposed a deep residual paradigm called the SRNet that minimized heuristic design elements in the system. It did not use any predefined high-pass filter as a preprocessing steo. It disabled the pooling step in the first convolutional blocks to prevent the information loss from weak stego signals. SRNet is currently one of the most powerful steganalysis methods for high-detection performance. However, it suffers from a large model size and high computational complexity. Zhang *et al.* [107] proposed the Zhu-Net that has 3×3 kernels and separable convolution operations to reduce the number of trainable parameters. The spatial pyramid pooling is utilized to enhance the representation ability of features through multi-level pooling. More recently, Reinel *et al.* [75] proposed the GBRAS-Net that combines the merits of SRN and Zhu-Net. GBRAS-Net performs well while maintaining a relatively small model size.

4.2.3 Green Learning

Green learning [49] aims to provide an energy-efficient and mathematically transparent solution to data-driven learning problems. GL-based models are modularized. They are trained in a feed-forward manner without back-propagation. Their training complexity is quite low that it can be carried out solely on CPU. Their model sizes are small and inference complexity is also modest. As a result, it is suitable for mobile/edge computing.

One key module in GL is "unsupervised representation learning". It exploits the underlying statistics of pixels to derive data-driven transforms such as the Saak transform [48] and the Saab transform [50]. Multi-stage Saab transforms can be cascaded to form a PixelHop system [14] for image classification. Distinct from DL-based models that determine filter weights by back-propagation, GL-based methods derive filter weights by analyzing the correlation structure of a local neighborhood centered at a pixel of interest.

Another key module in GL is "supervised feature learning" [93]. It is a feature selection strategy that chooses a more powerful subset of features by analyzing the entropy loss of each single feature dimension. Entropy loss is used to evaluate the discriminant power of each feature for the classification task. A feature subset with low entropy loss can offer the same or even better performance than the whole feature set. It can significantly reduce the feature dimension for the classifier, which in turn reduces the computational cost and the model size.

GL has been successfully applied to various vision tasks, such as image classification [14, 15], image anomaly localization [101], object tracking [109–112], image synthesis [2, 51–53, 114], and 3D point cloud classification, segmentation, and registration [41, 105, 106]. It has also been applied to image forensics such as deepfake video detection [10], GAN-generated fake image detection [115], etc.

In this work, we propose a novel GL-based image steganalyzer called GS. GS does not use heuristically designed high-pass filters as a preprocessor. It does not adopt the end-to-end optimized neural network, either. It differentiates stego and cover images based on anomaly detection of image patches. Although the stego signal is weak in patches, we can leverage pixel-based anomaly scores to zoom into fewer patches for further steganalysis. The proposed GS method is light-weight, mathematically transparent, and computationally efficient.

4.3 Green Steganalyzer (GS) Method

We present an overview of the proposed GS method and its rationale in Section 4.3.1. GS consists of three modules. They are detailed in Sections 4.3.2, 4.3.3, and 4.3.4, respectively.

4.3.1 Solution Overview and Rationale

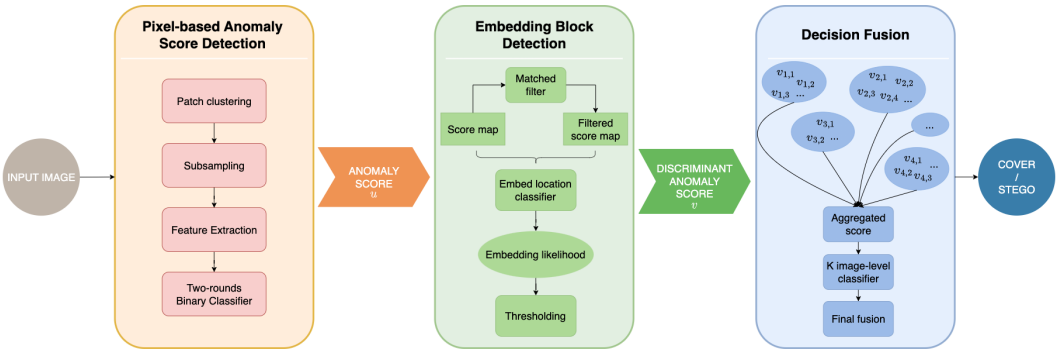


Figure 4.1: An overview of the proposed GS method.

For traditional steganalysis methods, one of the essential steps is to use predefined (or heuristically designed) filters to extract noise residuals from input images. It helps suppress image content and increase the signal-to-noise ratio (SNR). However, the fixed filter weights and a limited variety of filters cannot capture all of the complex cases in cover images and

embedded stego signals. DL-based steganalyzers can learn data-driven filters via backpropagation. They still use predefined high-pass filters in the pre-processing layer. Furthermore, they use a deep network architecture, leading to a large number of trainable parameters and heavy computational complexity.

To address the shortcomings of two prior methodologies, we propose a new pipeline and depict it in Figure 4.1. A high-level description of each module and its rationale are given below.

- Module 1: Pixel-based Anomaly Prediction

  Motivated by anomaly detection, we consider the stego embedding, especially embedding in smooth regions as 'anomaly'. We aim to predict the 'degree of anomaly' within local regions of images. The 'degree of anomaly' is named as 'anomaly score'. We first decompose an input image into overlapping image patches with stride equal to one. The high patch diversity makes any machine learning task challenging. To mitigate this problem, patches are grouped into multiple sets based on the embedding cost of each steganographic method. Then, patch diversity in a group is reduced. For each group, we train a binary classifier to discern two cases - anomaly patches that contain an embedded stego-signal (labeled by "1") and the corresponding raw patches from cover images (labeled by "0"). We obtain content-adaptive filters through unsupervised representation learning and the filter responses are used as features to the classifer in each group. We conduct an iterative classification scheme to enhance the detection performance, and derive an anomaly score for the central pixel of each patch.

- Module 2: Embedding Location Detection

  We design an embedding location detector to localize potential embedding locations based on anomaly scores. Since the anomaly score of a single pixel is noisy and untrustworthy, it is not reliable to use simple thresholding. We need a better idea.

That is, when a pixel is modified by a stegonagraphic scheme, it has an effect not only on its own anomaly scores also on those of its neighboring pixels. These pixels form an "anomaly spot". Then, the center of the anomaly spot is chosen as the embedding location. This strategy finds embedding locations more accurately.

- Module 3: Decision Fusion for Image-Level Classification

  We obtain the anomaly scores of all pixels in Module 1 and embedding pixel locations in Module 2. We aggregate anomaly scores of selected pixels from all groups for ultimate binary image-level decision (a cover or a stego image) in the last module.

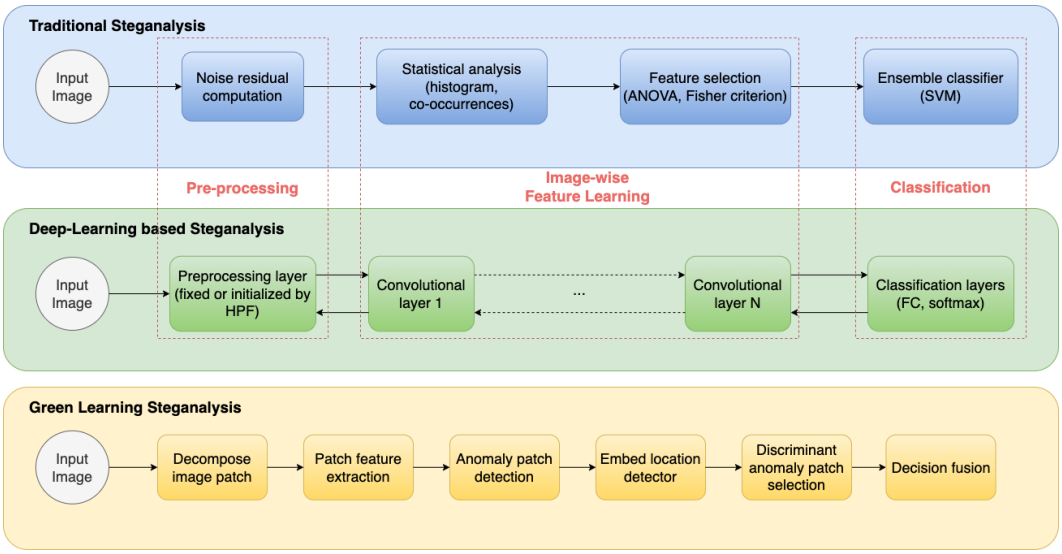


Figure 4.2: Comparison of the data processing pipelines: (top) traditional image steganalysis, (middle) DL-based image steganalysis, and (bottom) the GS method.

We show the data processing pipelines of traditional, DL-based, and the proposed GS steganalysis methods in Figure 4.2. There differences are summarized below.

1. Both traditional and DL-based steganalyers have a pre-processing step in the beginning of the pipeline. GS does not have this step. It derives an anomaly score of each pixel via supervised learning process.

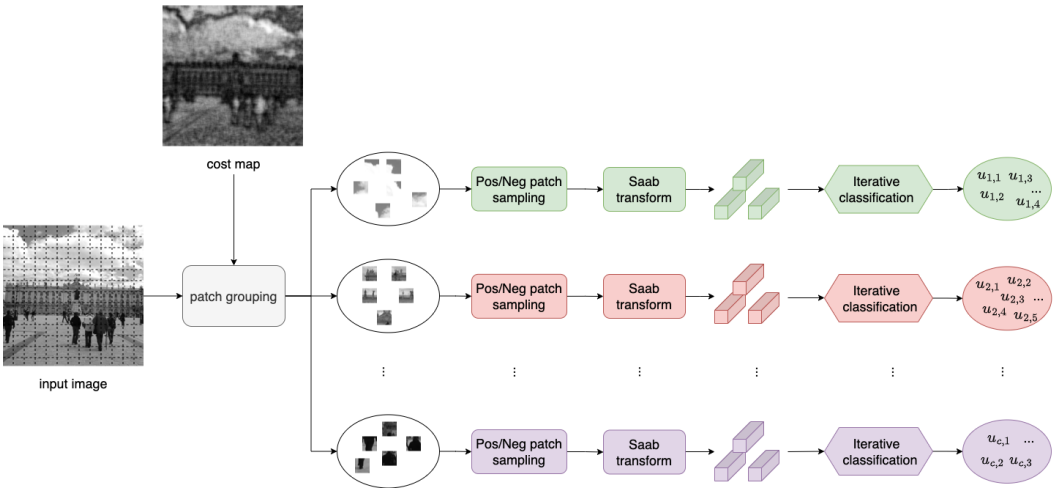


Figure 4.3: The block diagram of Module 1.

2. Both traditional and DL-based steganalyers examine image-level representations. In contrast, GL works on local regions only (either a patch in Module 1 or a neighborhood region in Module 2). This local processing can be parallelized easily. The memory requirement is much less.

3. For binary image-level classification, traditional steganalyers often leverages the ensemble of multiple classifiers (e.g., the ensemble of SVMs). The ensembled SVM classifier is slow in practice due to the high-dimensional features. DL-based steganalysis methods use the fully-connected (FC) layers and the softmax layers to make final decision. The number of trainable parameters in the FC layers is large. In contrast, GS uses the averaged anomaly score of anomaly spots as features to train multiple lightweight binary classifiers and an ensemble classifier. Its computation cost is much smaller.

4.3.2 Module 1: Pixel-based Anomaly Prediction

The block diagram of the first module is shown in Figure 4.3. The goal is to estimate the deviation of a patch of size $P \times P$ from its original one caused by steganographic embedding,

where P is a user-selected parameter. The anomaly caused by a stego signal in a smooth region is easier to tell than those in complicated textured regions. To address this challenge, we design a patch anomaly score predictor with the following four steps.

1. **Patch Grouping.** Input images are decomposed into overlapping patches of size $P \times P$ with stride 1. In the experiment, we set $P = 7$. Patches are grouped into multiple sets based on the embedding cost of a steganographic scheme. Each steganographic scheme has a specific way to compute the embedding cost of a pixel. If a pixel has a lower embedding cost, it has a higher embedding probability. Take S-UNIWARD steganography as an example. Its patch-wise cost varies from 0 to 11. We partition patches into groups with narrower cost ranges such as $[0, 1)$, $[1, 2)$, etc. Depending on the steganography algorithm, the group number ranges from 10 to 12.

2. **Positive and Negative Patch Sampling.** For each group, we select positive and negative patch samples in this step. We choose patches from stego images that contain at least one embedding bit and call them positive samples. Then, we find the corresponding patches from cover images and use them as negative samples.

3. **Feature Extraction.** We apply three Saab transforms to the patch center $[14, 50]$. The filter sizes are 3×3, 5×5, and 7×7, respectively. The Saab transform is a variant of the Principal Component Analysis (PCA) transform. It has two types of transform kernels: 1) the DC kernel, which gives the local average of pixels covered by the filter, and 2) AC kernels, which are data-driven kernels obtained by PCA. The reason to have two kernel types is that PCA can only be applied to zero-mean random vectors. By removing the local block mean, the block residual can be treated as a zero-mean random vector so that PCA can be applied. For a Saab transform of size $n \times n$, we have $n \times n$ filters. The filter responses are used as features since they describe the pixel local correlation structure. Compared with filters of smaller sizes, filters of larger

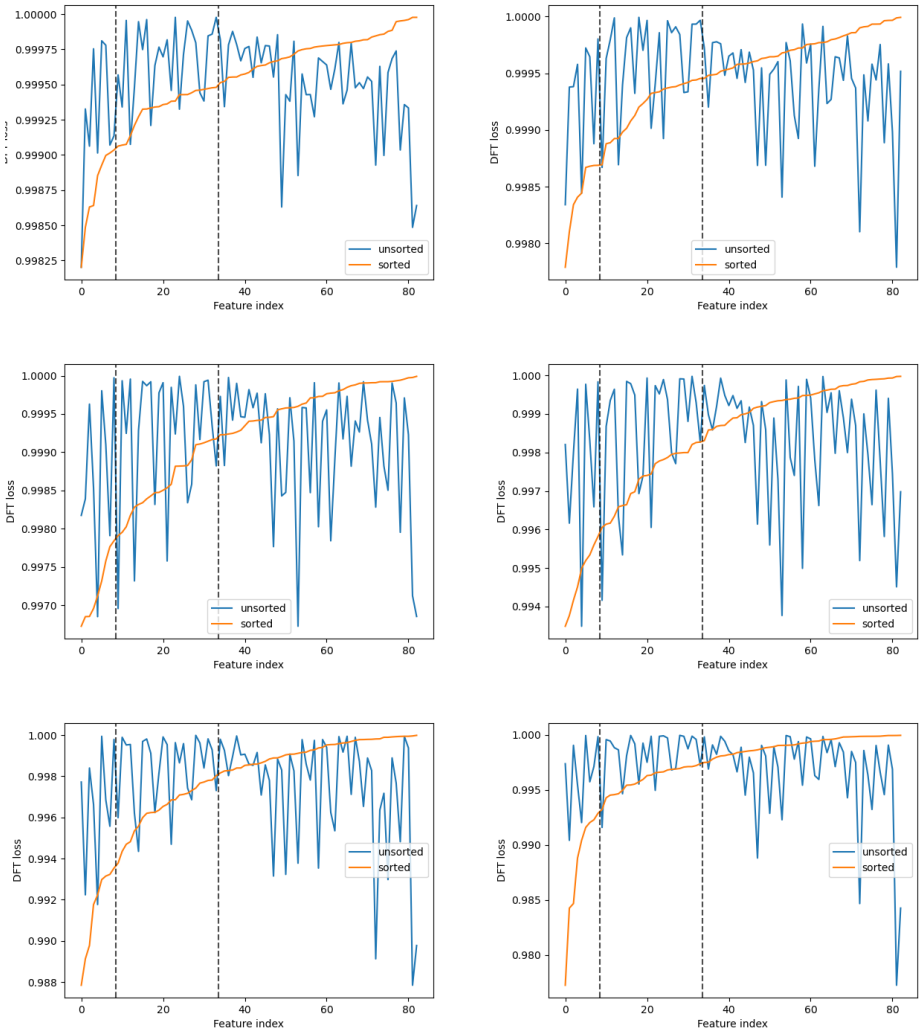


Figure 4.4: Discriminant feature test loss curves for some exemplar groups. From top to bottom, left to right, the group cost varies from low to high. For each subplot, blue curve represents *unsorted DFT loss, orange curve represents* sorted DFT loss. Vertical black dashed line indicates the 3 sets of features originating from 3×3 filters (left region), 5×5 filters (middle region), 7×7 filters (right region) respectively.

sizes are more discriminant in smooth regions but less in complicated regions. The aggregation of filter responses of different filter sizes is beneficial. By concatenating filter responses from three Saab transforms, we have $(3 \times 3) + (5 \times 5) + (7 \times 7) = 83$ features in total.

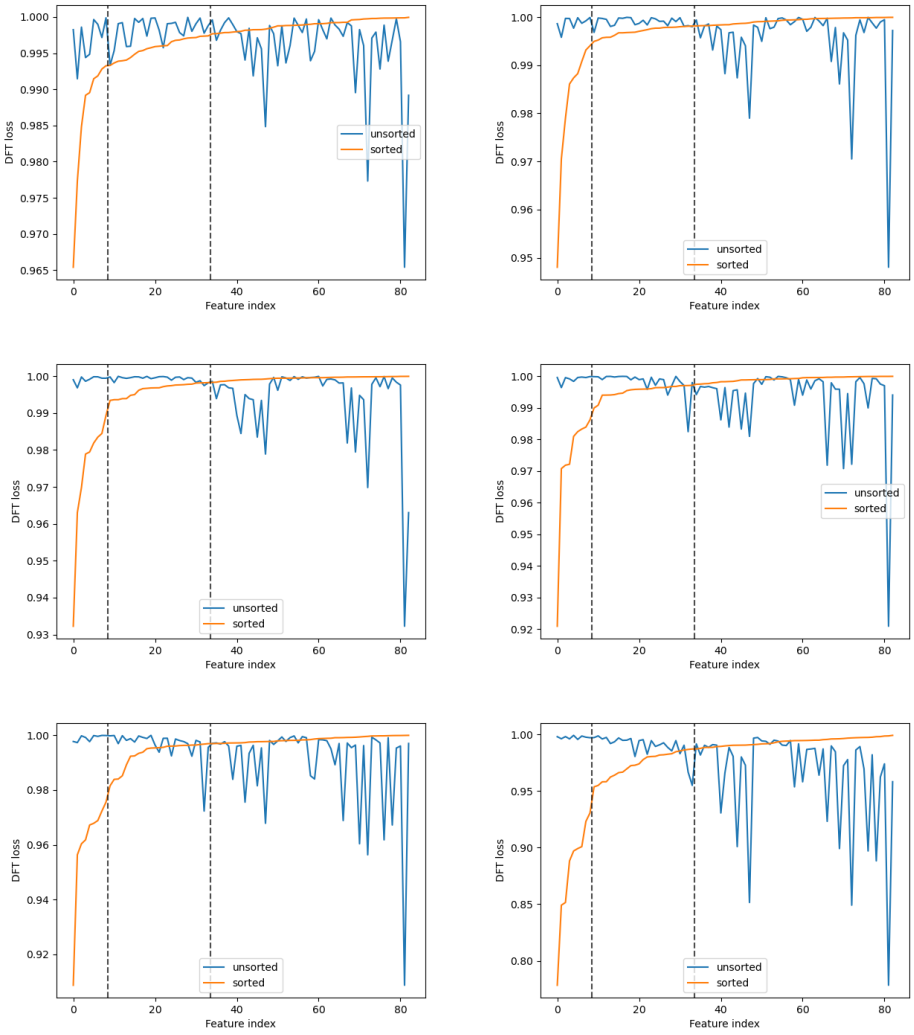


Figure 4.5: Discriminant feature test loss curves for some exemplar groups (cont.) From top to bottom, left to right, the group cost varies from low to high.

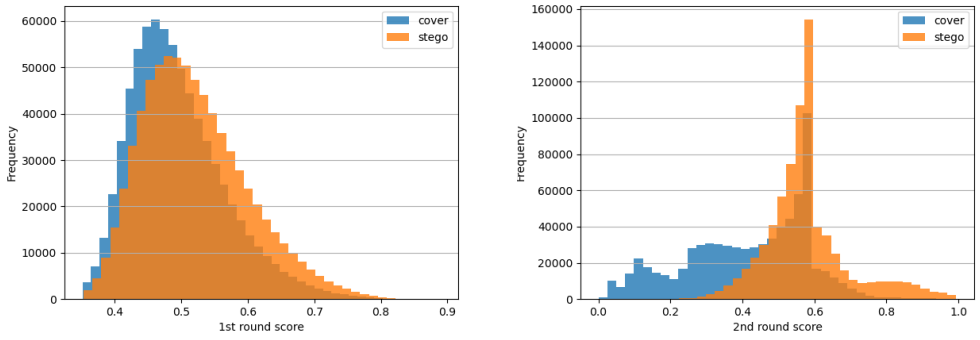
We select a subset of discriminant features using the discriminant feature test (DFT) [93] to reduce the feature dimension per group. The DFT calculates a loss value for each individual feature dimension. The DFT loss curves of four exemplar groups are shown in Figure 4.4 and 4.5, where their embedding costs increase from left-to-right and top-to-bottom. In each subplot, the blue and orange curves denote the original and sorted DFT losses, respectively. Two vertical dashed lines in black partition 83 unsorted features into 3 regions: features from 3×3 filters (the left region), 5×5 filters

(the middle region), 7×7 filters (the right region), respectively. Generally speaking, features from 7×7 filters are more discriminant while features from 3×3 filters are less discriminant. The gap of their discriminability widens as the embedding cost increases. To reduce the parameter number, we select 15 feature dimensions in all 10 groups. Selected feature are used to train a binary classifier to discern positive and negative patch samples collected in Step 2.

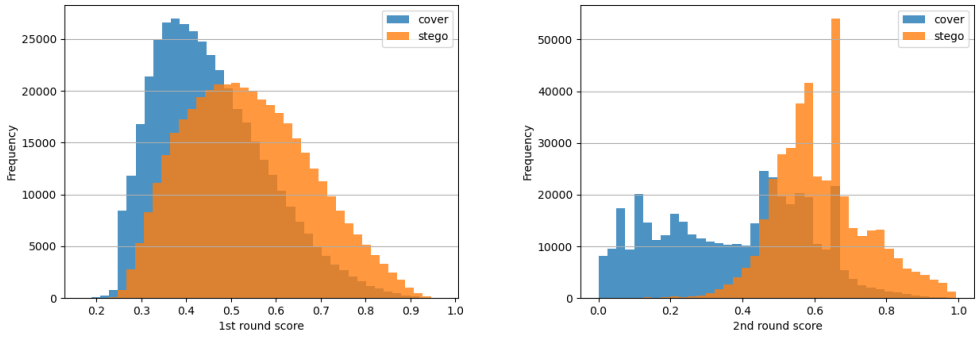
4. **Iterative Classification.** In the final step, we would like to obtain the anomaly score for each patch center using an iterative classification idea.

   i) Round-1 Classification. We first train a binary XGBoost classifiers [13] using labels and features obtained in Steps 2 and 3, respectively. In the inference stage, we keep the soft decision whose value lies between $[0, 1]$, which is called the round-1 anomaly score. If the score of a patch is closer to 0 (or 1), it is more likely to be drawn from cover (or stego) images. The predicted anomaly score distributions of positive and negative test samples (drawn from stego and cover images, respectively) are plotted in the left column of Figure 4.6 and 4.7. The figure has three rows. Each row indiciate a representative group as described in Step 1. It is observed that there is a significant overlap between positive and negative histograms. In words, they cannot be easily separated by one round of classification. To boost the classification performance, we adopt an iterative classification idea using the second-round classification.

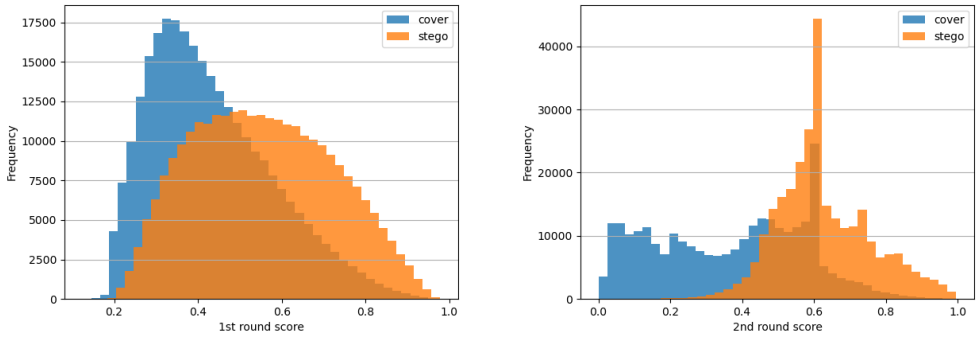
   ii) Round-2 Classification. For each group, we partition positive samples into subgroups accoording to their anomaly scores from Round 1. For example, we can divide their anomaly scores into ten uniform intervals, [0,0.1), [0.1, 0.2), ..., [0.9, 1.0]. Then, positive samples in the same interval form a subgroup. For each subgroup, we gather the corresponding negative samples to build training positive/negative pairs and use them to train the 2nd XGBoost classifier with their groundtruth labels (i.e., 1/0) by



(a)



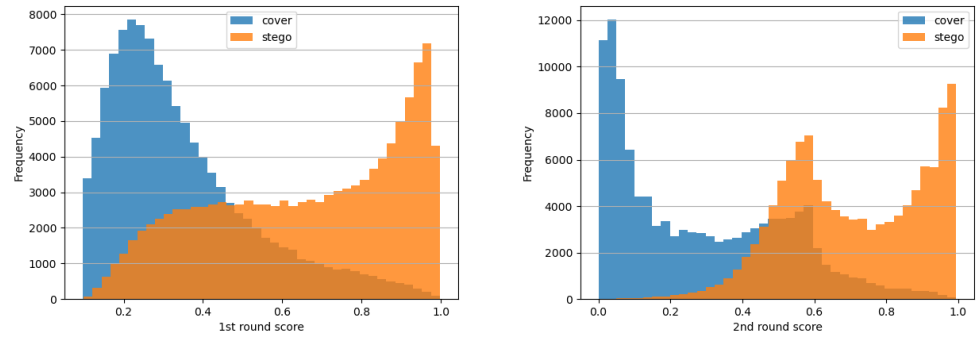
(b)



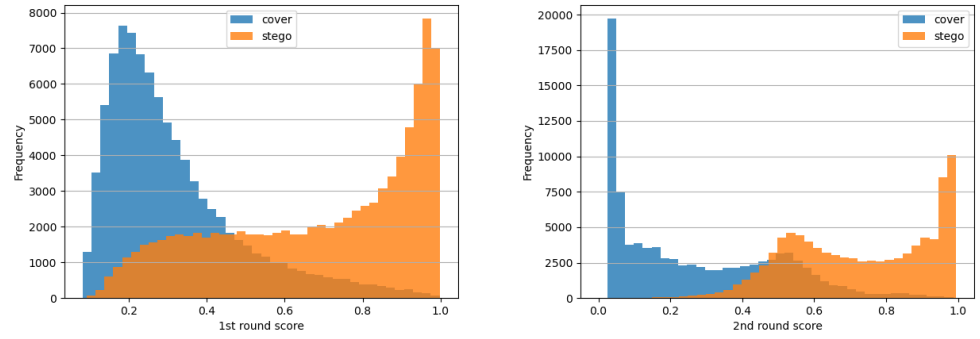
(c)

Figure 4.6: The anomaly score histograms of positive (in orange) and negative (in blue) test samples of three representative groups, where the left column and right column show results of the first- and the second-round XGBoost classifiers.

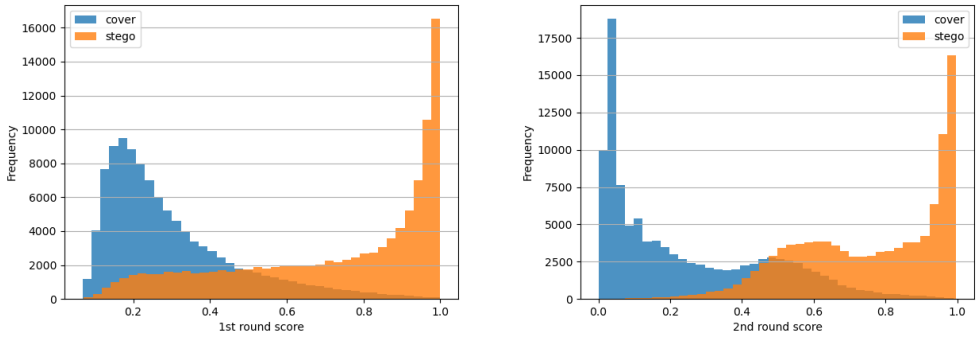
following Step 3 and Step 4.i. Then, we merge the predicted anomaly score distributions of positive/negative test samples from the 10 subgroups of the same group and plot them in the right column of Figure 4.6 and 4.7, where each row compares



(a)



(b)



(c)

Figure 4.7: The anomaly score histograms of positive (in orange) and negative (in blue) test samples of three representative groups, where the left column and right column show results of the first- and the second-round XGBoost classifiers (cont.).

the positive/negative histograms after Round-1 and Round-2 classifications for three selected representative groups.

We can see the clear advantage of the two-round iterative classification scheme by comparing the left and the right columns in Figure 4.6 and 4.7. First, the positive and negative histograms are more separated from each other. Second, the highest anomaly scores of stego patches from certain groups with round-1 classification can only reach 0.8 and 0.9 as shown in Figure 4.6(a), 4.7(a) and Figure 4.6(b), 4.7 (b), respectively. However, they can be boosted to 1.0 after Round-2 classification.

In the inference stage without image padding, we scan all interior pixels of test images using a window of size $P \times P$ with stride one, extract features using Step 3, and compute the anomaly score for the center pixel using Step 4.

4.3.3 Module 2: Embedding Location Detection

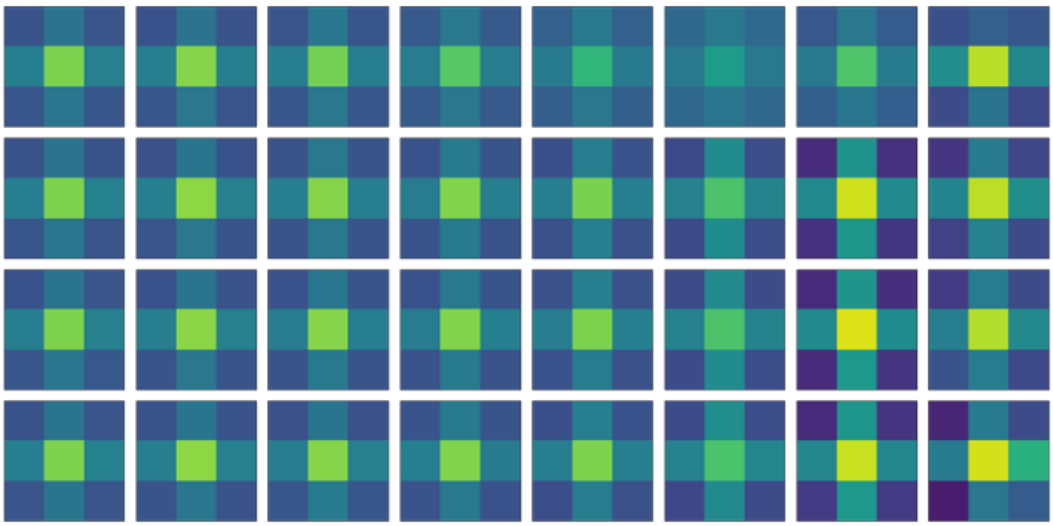


Figure 4.8: Visualization of matched filters from 8 different groups (i.e., one column per group) under the S-UNIWARD steganography algorithm with its payload equal to 0.4 bpp (1st row), 0.3 bpp (2nd row), 0.2 bpp (3rd row), 0.1 bpp (4th row). The brighter color indicates a larger value. The embedding cost increases from left to right in the same row (i.e. the same payload).

We get an anomaly score for each pixel and obtain an anomaly score map after Module 1. The purpose of Module 2 is to estimate the embedding location based on the anomaly

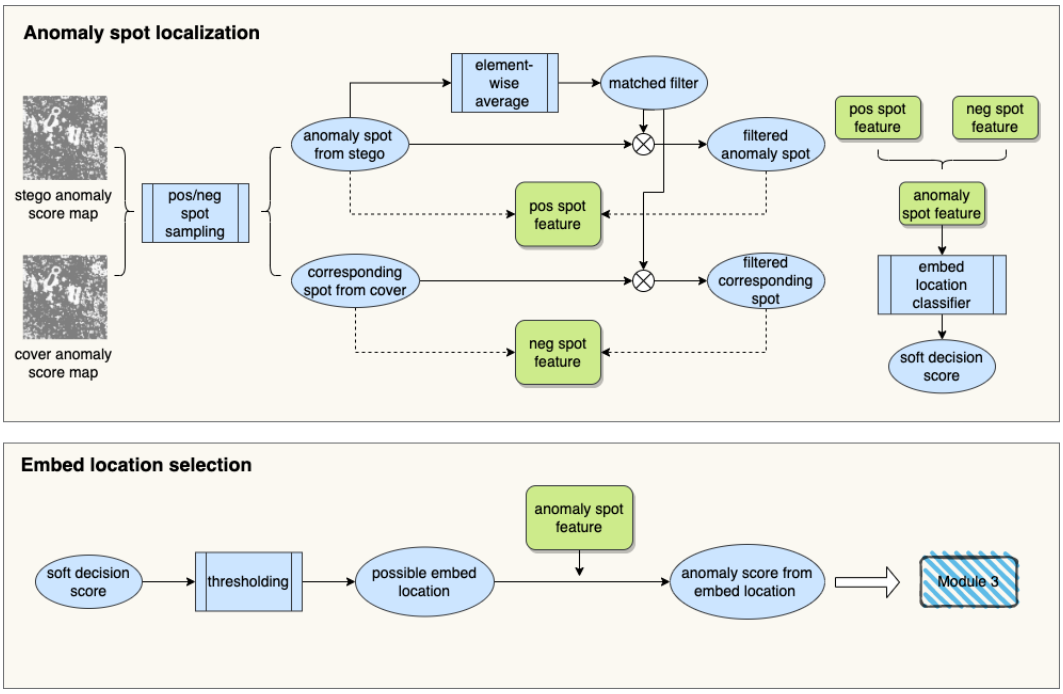


Figure 4.9: The block diagram of Module 2.

score map. Its block diagram is shown in Figure 4.9. It contains the following two major steps.

1. **Anomaly Spot Localization.** As shown in Figure 4.6, the anomaly score of a single pixel is still noisy, it is not reliable to use simple thresholding to decide the embedding location. Instead, we should consider a set of connected pixels jointly, which form an anomaly spot. Here, we set the size of anomaly spot to 3×3. We design an anomaly spot localizer for each individual group as follows.

   i) Positive and Negative Block Sampling. A block has a size of 3×3. We collect positive and negative block samples by following the same idea described in Step 2 of Module 1.

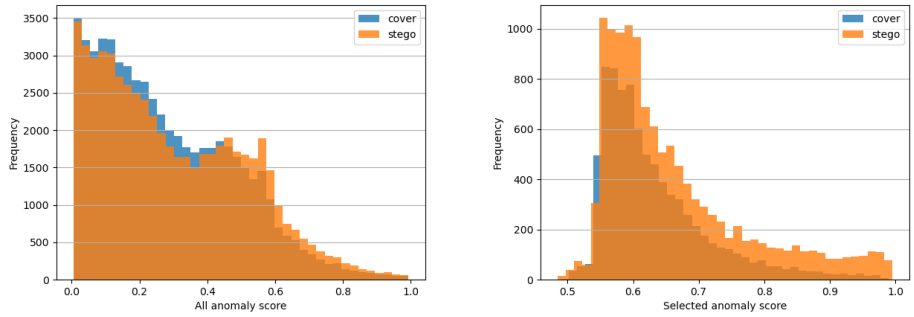
   ii) Matched Filtering for More Discriminant Features. The anomaly scores of nine pixels in a block of size 3×3 can be used as features to classify whether it is an anomaly spot through a binary classifier. However, these features are not discriminant enough

to ensure good classification performance. To boost the classification performance, we add another set of features using matched filtering. The matched filter is a 3×3 kernel. We collect the anomaly score maps of all positive samples and conduct element-wise averaging to obtain one matched filter for each group. The coefficient values of representative matched filters are visualized in Figure 4.8. All of them share similar properties. First, its central coefficient has the largest value, four sides' coefficients are smaller, four corners' coefficients are the smallest. The values of four sides are close to each other, and the values of four corners are also close to each other. This can be explained by the symmetrical property in space. Second, the range of coefficients is wider for a higher embedding cost. The application of matched filtering to the anomaly score map of a block will enhance the difference between positive and negative samples.

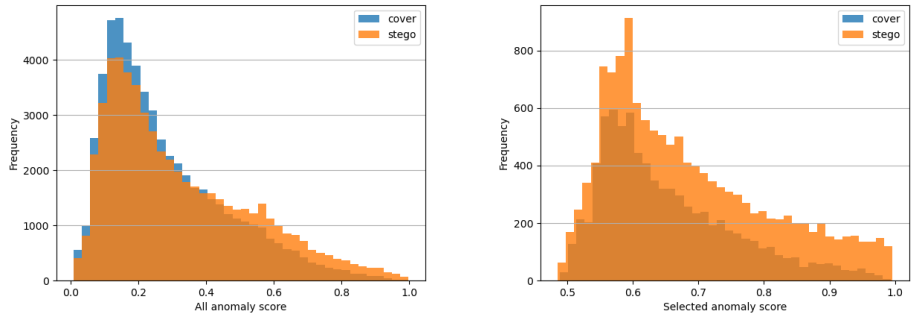
iii) Classification. We apply matched filters to anomaly maps to get pixel-wise matched-filter responses for each group. A block has 9 pixels, and each pixel has its own anomaly score and matched-filter response. Then, we use 9 anomaly scores and 9 matched-filter responses to form an 18-D feature vector, train an XGBoost binary classifier. The soft decision score indicates the likelihood for a block to contain embedding bits.

2. **Embedding Location Selection.** If the soft decision score of a block is higher than a threshold, it serves as a candidate for consideration in the image-level decision. It is called an anomaly spot. The threshold is selected by optimizing the F1 score, which is a measure used to balance false positives and false negatives. The center of the anomaly spot is the detected embedding location.

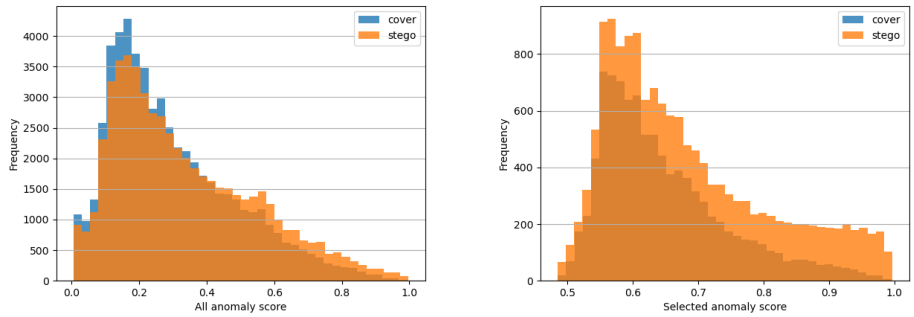
To show the importance of Module 2, we compare the distributions of anomaly scores from Module 1 and distributions of soft decision scores of anomaly spots from Module 2 for three representative images in Figure 4.10 and 4.11. First, we want to point out that the



(a)



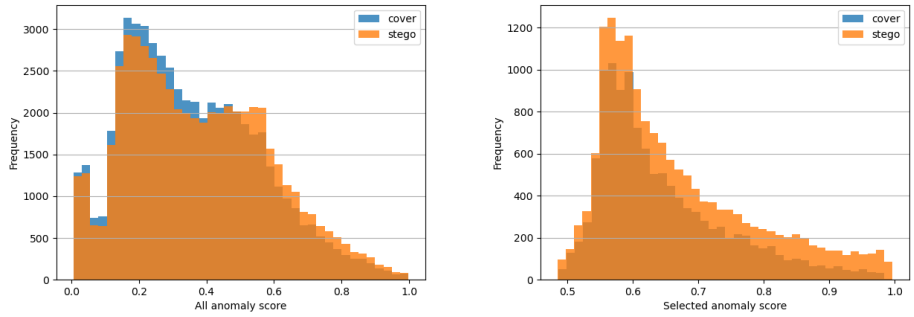
(b)



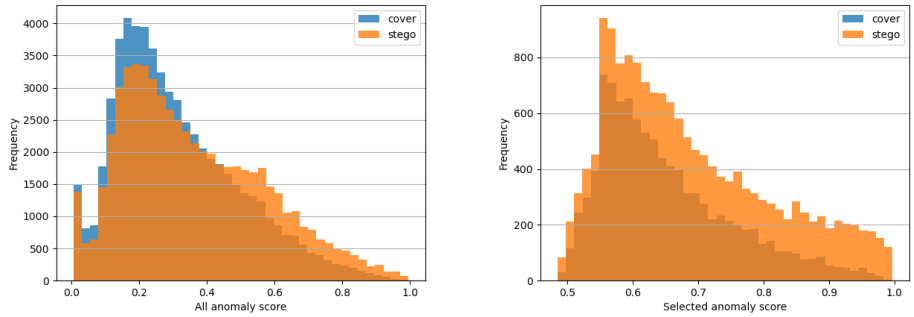
(c)

Figure 4.10: Comparison of distributions of anomaly scores from Module 1 (in the left column) and distributions of soft decison scores of anomaly spots from Module 2 (in the right column) for three representative images (in three rows).

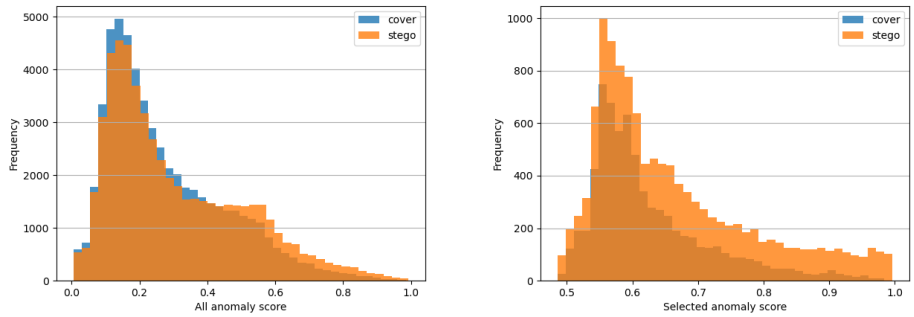
number of anomaly spots is significantly less than the number of pixels in test images. Thus, a large number of pixels are already removed in Module 2. Second, we should pay special attention to the right tail of the histogram. Comparing with the distribution Module 1, the distribution in Module 2 is more separable in the right-tail region. The discriminant ability



(a)



(b)



(c)

Figure 4.11: Comparison of distributions of anomaly scores from Module 1 (in the left column) and distributions of soft decison scores of anomaly spots from Module 2 (in the right column) for three representative images (in three rows) (cont.).

of positive/negative samples after Module 1 is still weak. Yet, they are more distinguishable after Module 2.

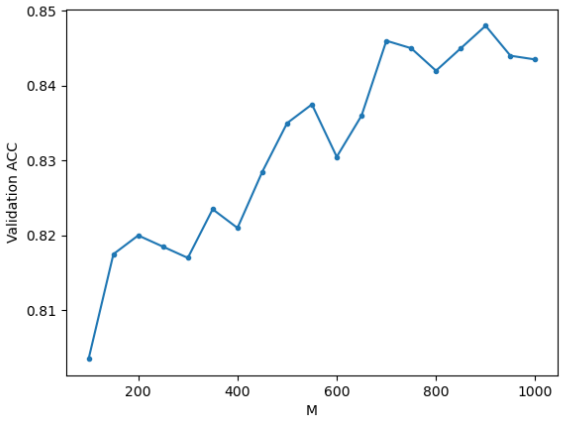


Figure 4.12: The accurate classification rate as a function of M values applied to the validation dataset.

4.3.4 Module 3: Decision Fusion for Image-Level Classification

Recall that we obtain the anomaly scores for all pixels in Module 1 and select embedding pixel locations in Module 2. In Module 3, we aggregate anomaly scores of selected pixels from all groups. We sort their anomaly scores from the highest to the lowest, and use top M anomaly scores as features for image-level decision. If M is too small, the decision is not reliable. If M is too large, we may include unrelable pixels in the decision. Thus, a proper value of M has to be determined based on the validation dataset. We conduct a grid search for the optimal M values from 100 to 1000, with step size 50. We show the accurate classification rate as a function of M for a representative validation image. We see that the accuracy goes higher as M becomes larger. However, it is not an monotonically increasing curve. In the experiment, we choose five M values, M_i, $i = 1, \cdots, 5$ and conduct an XGBoost classifier for each M value. Each XGBoost will give a binary decision - stego or cover. The final image-level decision is made by the majority vote.

4.4 Experiments

We conduct experiments to demonstrate the effectiveness and efficiency of the proposed GS scheme in this section. The experimental setup is first described in Sec. 4.4.1. Then, we compare the effectiveness of GS with 7 benchmarking methods in terms of the detection error rate in Sec. 4.4.2. Finally, we compare the efficiency of GS with 5 benchmarking methods by examining their model sizes and computational complexity in Sec. 4.4.3.

4.4.1 Experimental Setup

The settings of our experiments are given below.

Dataset. Experiments are conducted on the BOSSbase v1.01 dataset [3]. It contains 10,000 8-bit gray-scale images of resolution 512×512. They are stored of the uncompressed Portable Gray Map (pgm) format. These images are acquired with several cameras. They cover diverse natural scenes with various texture characteristics. The BOSSbase dataset has been widely used as a test dataset for digital image staganalysis. We split 10,000 BOSSbase images evenly into 50% training data and 50% testing data. For fair comparison with other benchmarking methods in [74], [91], [94], [107], we resize raw images of resolution 512×512 to new images of resolution 256×256 in both training and test datasets and evaluate the GS method on resized images.

Steganographic Schemes. We consider S-UNIWARD, WOW and HILL three content-adaptive steganographic schemes and implement them using the Syndrome-Trellis Codes (STC). For each steganography scheme, stego images are generated with two payloads - 0.2bpp and 0.4bpp. In steganography, the less the payload, fewer bits are embedded, making stego images more difficult to detect.

Benchmarking Methods. We compare GS with the following representative steganalysis methods:

- Traditional Methods: Spatial Rich Model with Ensemble classifier (SRM+EC) [30];
- Earlier DL-based Methods: QianNet [74], Xu-Net [91], Ye-Net [94], and Yedroudj-Net [96];
- Recent DL-based Methods: Zhu-Net [107] and GBRAS-Net [75].

All of them are tested under the same train/test split as ours.

Evaluation Metrics. We use the averaged detection error rate,

$$P_E = \frac{1}{2}(P_{FA} + P_{MD}), \tag{4.1}$$

as the performance evaluation metric, where P_{FA} and P_{MD} are the false alarm probability and the missed detection probability, respectively. Besides detection accuracy of steganalyzers, we also examine their model sizes measured by the number of model parameters and computational complexity measured by the number of floating-point operations (FLOPs) in the inference stage.

4.4.2 Detection Performance Evaluation

First, we compare detection error rates of GS against 7 benchmarking methods on the S-UNIWARD and WOW datasets with payloads of 0.2 bpp and 0.4 bpp in Table 4.1. The best and second best results are highlighted in bold and with an underline, respectively. As shown in the table, the two recent DL-based methods (i.e. Zhu-Net and GBRAS-Net) and our GS rank the top three. Among the three, GBRAS-Net has the best performance for both S-UNIWARD and WOW steganographic embeddings. GS achieves the second best for S-UNIWARD and the third best for WOW. Zhu-Net ranks the third for S-UNIWARD and the second for WOW.

Table 4.1: Comparison of detection error rates (P_E) against S-UNIWARD and WOW steganographic schemes at paylods equal to 0.2 bpp and 0.4 bpp, where the best is in bold and the second best is underlined.

Method	Payload			
	S-UNIWARD 0.2 bpp	S-UNIWARD 0.4 bpp	WOW 0.2 bpp	WOW 0.4 bpp
SRM+EC	36.6	24.7	36.5	25.5
Qian-Net	46.3	30.9	38.6	29.3
Xu-Net	39.1	27.3	32.5	20.7
Ye-Net	39.9	31.3	33.1	23.3
Yedroudj-Net	36.5	22.6	27.7	14.9
Zhu-Net	28.6	15.5	<u>23.1</u>	<u>11.9</u>
GBRAS-Net	**26.4**	**12.9**	**19.7**	**10.2**
GS (Ours)	<u>27.86</u>	<u>13.63</u>	24.05	12.18

Table 4.2: Comparison of detection error rates (P_E) under the HILL steganography at paylods equal to 0.2 bpp and 0.4 bpp, where the best is in bold and the second best is underlined.

Method	Payload	
	HILL 0.2 bpp	HILL 0.4 bpp
Zhu-Net	33.4	23.5
GBRAS-Net	**31.5**	**18.1**
GS (Ours)	<u>33.13</u>	<u>23.29</u>

Since no error rates are reported by Qian-Net, Xu-Net, Yedroudj-Net and Zhu-Net for the HILL steganography, we only compare GS with Zhu-Net and GBRAS-Net with payloads equal to 0.2 bpp and 0.4 bpp for HILL in Table 4.2. This is sufficient since GS, Zhu-Net, and GBRAS-Net are the top three performers in Table 4.1. Again, we observe that GS and Zhu-Net have comparable performance and their performance is inferior to that of GBRAS-Net.

4.4.3 Model Sizes and Computational Complexity

Model sizes. We compare the model sizes of GS and three other DL-based steganalyzers in Table 4.3, where the model size is defined as the number of trainable parameters. The trainable parameters of GS include Saab filter parameters and anomaly patch classifier parameters in Module 1, embedding location classifier parameters in Module 2, and decision

fusion classifier parameters in Module 3. For a given steganography scheme, we partition patches and blocks into 10 groups based on its embedding cost. The parameter number of each module is calculated below.

1. Module 1

   There are three parts: Saab filter banks and XGBoost classifiers.

   a) Saab Filter Banks. In each group, we train with three Saab filter banks of size 3×3, 5×5, and 7×7. As mentioned in Section 4.3.2, we select 15 filters among the 83 filters. Among the 15 selected filters, we count the number of filters from 3×3, 5×5, 7×7, respectively, and plot the bar plot in Figure 4.13. Different groups have different sets of selected filters. They are however consistent for all three experimented steganography algorithms. Based on the statistics in Figure 4.13, we aggregate the Saab parameters from all groups and have $7,364$ parameters. For different steganography algorithms, we can safely say that our Saab parameters is no more than $8k$. Thus, the number of Saab parameters for all 10 groups is reduced from $(3{\cdot}3{\cdot}9{+}5{\cdot}5{\cdot}25{+}7{\cdot}7{\cdot}49){\cdot}10 = 31,070$ to $8K$ in Module 1.

   b) XGBoost Classifiers. For iterative classifiers in Module 1, we use the XGBoost classifier with 100 trees and maximum depth of 2. Each tree has a maximum depth of 2 so that it has at most 10 parameters. There are approximately $1K$ parameters per XGBoost classifier. The first round has 10 classifiers (one for each of the 10 groups). The second round has $10 \times 10 = 100$ classifiers. There are 110 XGBoost classifiers with $110K$ parameters in total.

   By combining (a) and (b), the total number of parameters in Module 1 is equal to $8K + 110K = 118K$.

2. Module 2

   We need 10 XGBoost classifers (one for each of the 10 groups). They have the same

hyper-parameters as those in Module 1. Thus, the number of parameters is $1K \cdot 10 = 10K$.

3. Module 3

   As mentioned in Sec. 4.3.4, we need 5 classifiers with M_i features, $i = 1, \cdots, 5$. Thus, the total number of parameters in Module 3 is $1K \cdot 5 = 5K$.

Then, the total number of parameters of the GS method is $118K + 10K + 5K = 132K$.

We measure the model sizes of 3 high-performance DL-based steganalyzers and list them in Table 4.3. Among them, Yedroudj-Net and Zhu-Net have larger model sizes because of deep-layer architectures and denser FC layers. GS has the smallest model size. It is typical to assign 4 bytes to each parameter. Then, the GS model demands 528K byte memory.

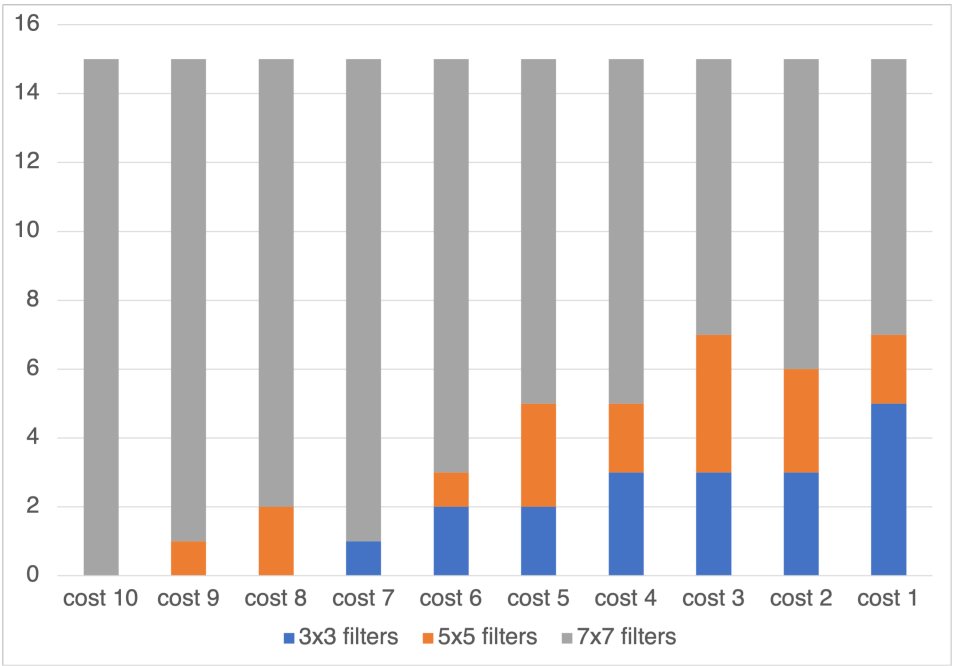


Figure 4.13: Comparison of selected channel distributions for 10 groups based on embedding costs, where "cost 1" denotes the lowest embedding cost group, "cost 10" denotes the highest embedding cost group, and blue, orange, and gray denote filters of size 3×3, 5×5, and 7×7, respectively.

Computational Complexity. We measure the number of floating-point operations (FLOPs) per pixel in the *inference* stage as an indicator of computational complexity. There

Table 4.3: Comparison of model sizes and computational complexities of 6 steganalyzers, where we use "X" to demonstrate the ratio of numbers with respect to the reference (denoted by 1X).

Method	Number of parameters	KFLOPs/pixel
Yedroudj-Net	252,459 (1.91X)	190.73 (54.03X)
Zhu-Net	277,156 (2.10X)	45.62 (12.92X)
GBRAS-Net	166,598 (1.26X)	90.79 (25.72X)
GS (Ours)	132,000 (1.00X)	3.53 (1.00X)

are several APIs in PyTorch or Keras to measure FLOPs. The numbers of FLOPs for the DL-based methods in Table 4.3 are measured using the *keras-flops* package for each image and then divided by $256 \cdot 256$, which is the total number of pixels per image.

Since the GS model is not a neural-network-based model, the *keras-flops* package cannot be directly used. Instead, we compute its number of FLOPs analytically as follows.

1. Module 1

   There are three parts: patch cost calculation, Saab filter banks and XGBoost classifiers. *a) patch cost.* For each pixel location, there needs 8 additions and 1 division, resulting in 9 FLOPs.

   a) Saab Filter Banks. For Saab filters of size $n \times n$, the total number of operations per filter is about $2 \times n \times n$ since the inner product of two 9-D vectors involve 9 multiplications and 8 additions. There are three Saab filter sizes. The number of FLOPs for all three Saab filters per pixel is equal to $2\times(3\times3\times9+5\times5\times25+7\times7\times49) = 6214$. As explained in both Section 4.3.2 and Section 4.4.3 model size part, we select 15 filters among all of them. Based on Figure 4.13, we calculate number of FLOPs for each group and sum them up to be $2,512$. Thus, the FLOPs/pixel number is actually reduced to $2,512$.

   b) XGBoost Classifiers. We conduct Round-1 and Round-2 two XGBoost classifiers at each pixel location. The computational complexity of an XGBoost is a subtraction at each node and one sample will trace only one path. Thus, the complexity for all

trees is the tree depth multiplied by the tree number and number of classes. All trees prediction need to be summed up via addition. Thus, the total complexity is equal to $2 \times 100 \times 2 + 100 = 500$.

By combining (a) and (b), the FLOPs/pixel number in Module 1 is $2512+500 = 3,012$.

2. Modules 2

   Module 2 computation involves the convolution with matched filters and XGBoost classification. Since we choose anomaly spot size as 3×3, for each pixel location, convolution needs 9 multiplications and 8 additions. As for XGBoost classifier, the number of FLOPs/pixel is $2 \times 100 \times 2 + 100 = 500$. Thus, the total FLOPs/pixel number in Module 2 is 517.

3. Modules 3

   There are five XGBoost classifiers in Module 3 per image. Thus, the number of FLOPs is equal to $5 \times (2 \times 100 \times 2 + 100) = 25K$. We need to divide this number by 250×250 pixel locations per image since no image padding is used. The number of FLOPs/pixel is equal to 0.4, which is neglible as compared to the numbers of FLOPs/pixel in Modules 1 and 2.

The total number of FLOPs/pixel of GS is about $3,012 + 517 = 3,529$, which is far less than that of benchmarking DL models.

4.5 Conclusion and Future Work

A GL-based image steganalysis method, called Green Steganalyzer (GS), was proposed in this work. GS is a lightweight modularized image steganalysis method. It contains three modules. First, it assigns an anomaly score to a center pixel of a patch. Next, it studies the relationship of anomaly scores between a pixel and its neighbors to estimate the embedding

likelihood of the center pixel. Finally, it selects pixels of higher embedding probabilities and conducts decision that error detection rates of GS are competitive with state-of-the-art DL-based steganalyzers against S-UNIWARD, WOW and HILL three steganographic schemes. At the same time, it demands a smaller model size and lower computational complexity than DL-based methods. Furthermore, GS is mathematically transparent due to its modular design.

As for future extensions, we would like to test GS on the ALASKA 2 dataset. It is a more challenging dataset than BOSSbase v1.01 since it contains more natural scenes and images of various resolutions. DL- based steganlyzers are restrained on a certain input image size because of their architecture design. In contrast, our GS model can handle different image sizes between training and testing dataset. Also, it is desired to improve the detection performance of GS furthermore with slightly higher complexity.

Chapter 5

Conclusion and Future Work

5.1 Summary of the Research

In this dissertation, we focus on two tasks of image forensics: GAN-generated image detection and image steganalysis. For both works, we provide green (light-weight), high-performance and mathematically transparent solutions to deal with the tasks.

In the first work, we propose RGGID, which is a robust, green GAN-fake image detector. We take advantage of the assumption that GAN architectures usually fail to synthesize well on high-frequency components of images, such as high-quality details, complex textures, edges, etc. We decompose image into small blocks and select ones from complex regions since they are more discriminant in terms of high-frequency components. By utilizing unsupervised feature learning method, PixelHop, we extract feature of blocks in a computationally efficient way. We measure the discriminant ability of each feature channel of blocks by training classifier on it, and select classification soft decision from more discriminant ones. By designing the two-end decision fusion strategy, image-wise classification is made by ensemble classifier based on fused feature. RGGID offers a green solution for GAN-generated image detector since its model size is significantly smaller than that of deep neural networks (DNNs). We apply common manipulations to real/fake source images, including

JPEG compression, resizing and Gaussian additive noise, and demonstrate the robustness of RGGID to these manipulations. Furthermore, we demonstrate the generalization ability of RGGID on 11 unseen generative architecture and dataset by training solely on ProGAN image and testing on other 11 dataset.

In the second work, we propose Green steganlyzer(GS), which is a green steganalysis method for detecting content-adaptive spatial steganography. Different from traditional steganalysis and deep learning-based steganalysis methods, GS does not contain any heuristically designed components and does not need end-to-end training. It contains three modules. First, it assigns an anomaly score to a center pixel of a patch. Next, it studies the relationship of anomaly scores between a pixel and its neighbors to estimate the embedding likelihood of the center pixel. Finally, it selects pixels of higher embedding probabilities and conducts decision that error detection rates of GS are competitive with state-of-the-art DL-based steganalyzers against S-UNIWARD, WOW and HILL three steganographic schemes. At the same time, it demands a smaller model size and lower computational complexity than DL-based methods. Furthermore, GS is mathematically transparent due to its modular design.

5.2 Future Research Topics

The proposed green learning methods in image forensics applications are proved advantageous in binary classification tasks like GAN-fake image detection and image steganalysis. We are interested in exploring its potential in more complicated image manipulation scenarios. We bring up the following research problems:

- **Image Splicing Localization**. This is a localization task which requires localization of areas of spliced in spliced images. Given an image with some spliced region, can we output a mask of spliced region over authentic region?

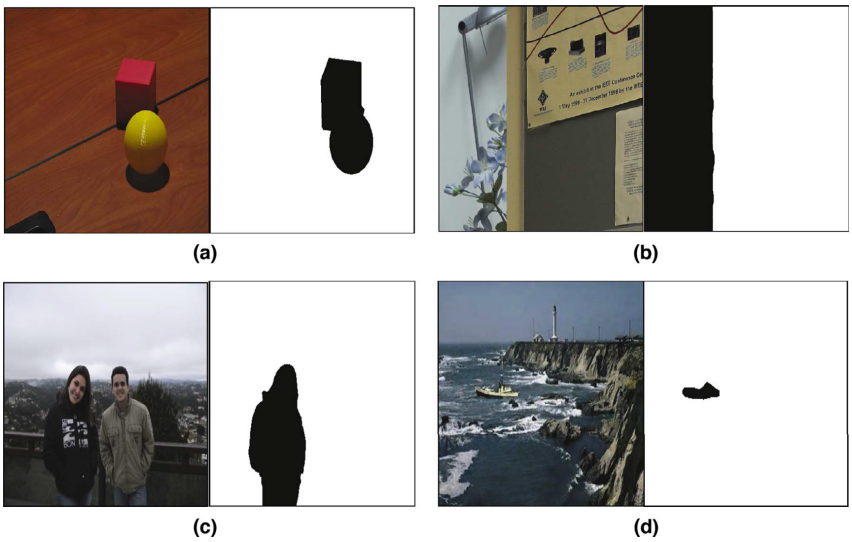


Figure 5.1: Examples of spliced images and corresponding ground truth masks from four different datasets: (a) the Nimble Science (SCI) dataset, (b) the Columbia Uncompressed dataset, (c) the Carvalho dataset, and (d) the CASIA v1.0 dataset. For the ground truth mask, pixels that were manipulated are represented by a value of 0 (the black region) and pixels that were not manipulated are represented by a value of 255 (the white region).

- **Image Forgery Detection**. This is a detection task which requires classification of images as forged or pristine (never manipulated). Given an image, it may include manipulations like splicing, copy-move, in-painting, or combinations of them, can we classify whether it's forged or pristine?

5.2.1 Image Splicing Localization

Image splicing is one of the most common type of image forgery. It manipulates images by copying a region from one image (the donor image) and pasting it onto another image (host image). The output is often called spliced images. Image splicing forgery is often used to give false impression to the audience or potentially used to generate false agenda for political purposes. With the advent of image manipulation techniques on the web, image splicing localization remains an interesting yet challenging topic nowadays. Figure 5.1 shows 4 example spliced images are their corresponding ground truth masks.

Many splicing localization techniques are proposed by researchers and they can be roughly divided into three classes based on the pattern or trace types used to separate the spliced region from the rest of the image. They exploit the following traces (or features): (1) noise patterns, (2) Color Filter Array (CFA) interpolation patterns, and (3) JPEG-related traces.

All three classes focus on exploiting the statistical difference between spliced region and remaining region of host image. The first class [11, 17, 20, 55, 65, 67] exploits noise patterns under the assumption that different images have different noise patterns as a result of a combination of different camera makes/models, the capture parameters of each image, and post-processing techniques. Since the spliced region originated from a different image (the donor image) than the host image, the spliced region may have a noise pattern that is different than the noise pattern in the remaining region of the host image. The second class of algorithms [23, 25] exploits CFA interpolation patterns. Most digital cameras acquire images using a single image sensor overlaid with a CFA that produces one value per pixel. CFA interpolation is a process to reconstruct the full color image by transforming the captured output into three channels (RGB). Splicing can disrupt the CFA interpolation patterns in multiple ways. Thus, it can be exploited to localize spliced regions. The third class of algorithms exploits the traces left by JPEG compression [1, 4–6, 24, 58, 60, 64, 95]. Most of these methods use features from JPEG quantization artifacts or JPEG compression grid discontinuities. Original image are assumed to undergo consecutive JPEG compression, while the spliced portion may have lost its initial JPEG compression characteristics due to smoothing or resampling of the spliced portion. These incongruous features can help localize a spliced region.

Back in 2017, Salloum *et al.* proposed a splicing localization method based on fully convolutional neural networks (FCN) [62]. Motivated by the coarse localization output of single-task FCN (SFCN), they propose the use of a multi-task FCN (MFCN) [80] that

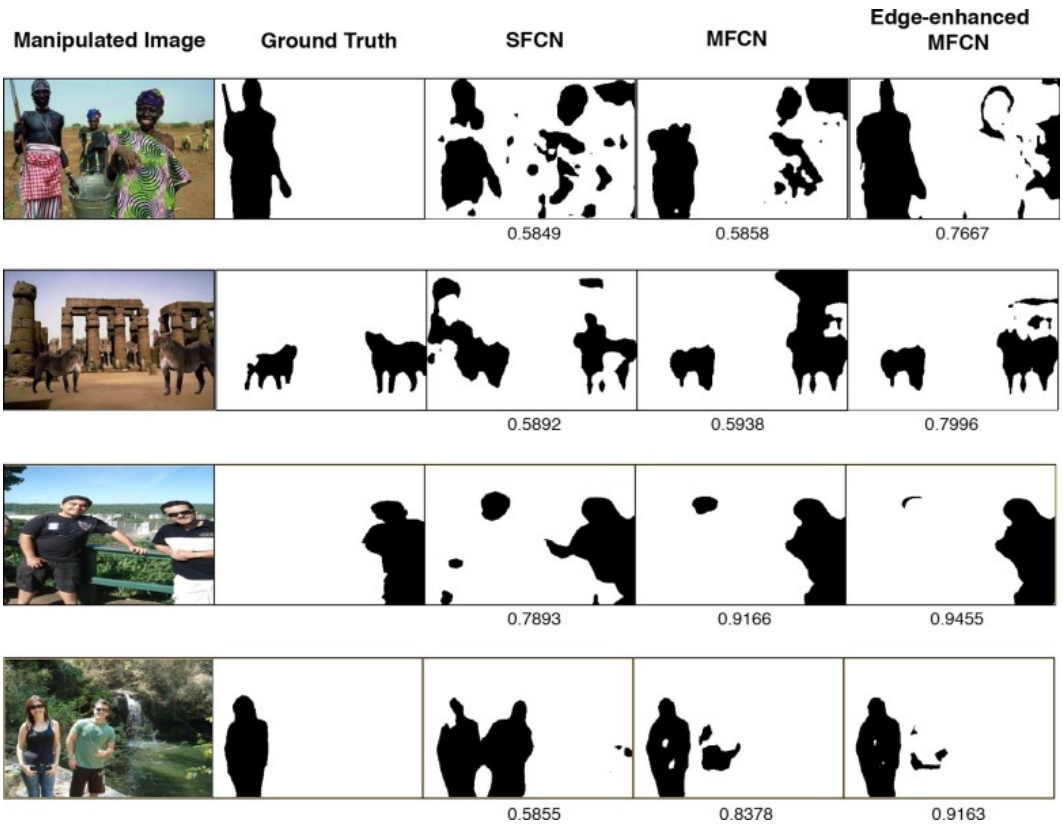


Figure 5.2: Output mask examples of SFCN, MFCN and edge-enhanced MFCN methods

utilizes two output branches for multi-task learning. One branch is used to learn the surface label, while the other branch is used to learn the edge or boundary of the spliced region. Output mask examples are showed in Figure 5.2. The number below each output example is the corresponding F1 score. MFCN is a successful splicing localization solution and it can provide finer localization output.

Different from neural-network based methods, we plan to solve this problem based on green learning methodology. Since we are not training the system in end-to-end manner, pixel-wise classification decision has to be made and form an output spliced region mask. Preliminary design of our method is showed in Figure 5.3. We will still use unsupervised feature learning method, PixelHop++ unit to extract features within certain neighborhood

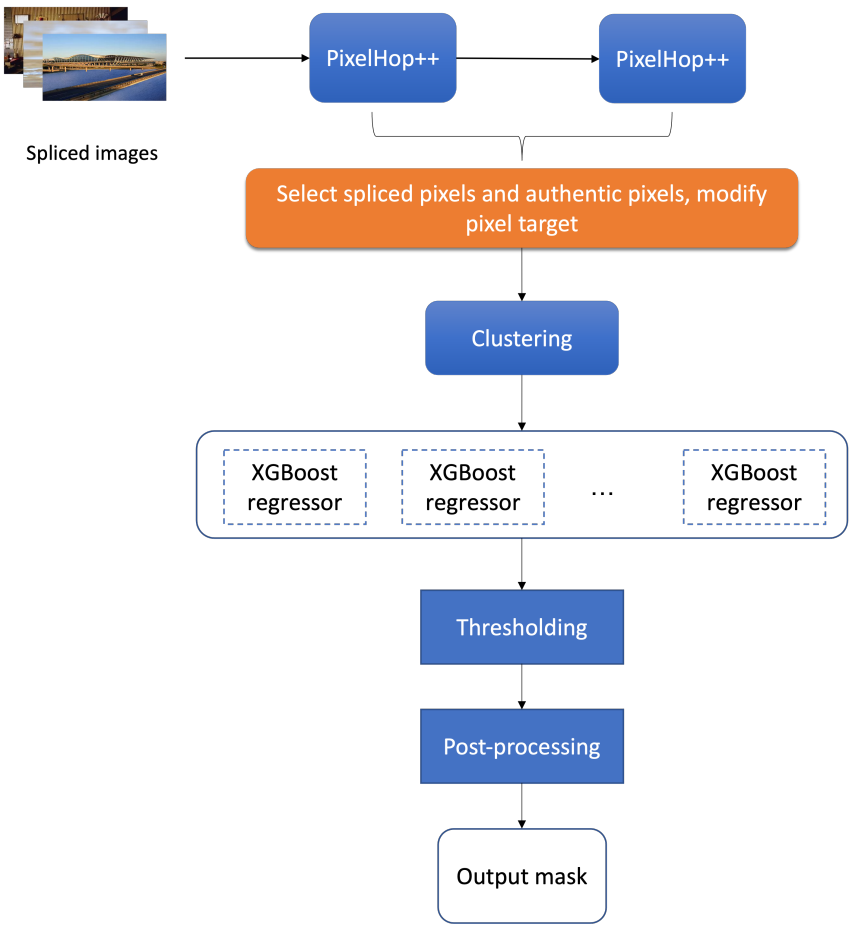


Figure 5.3: Preliminary design of green learning based approach for image splicing localization.

of center pixel. Note that the spatial dimension is preserved after two PixelHop++ units, so that we can make decision on all pixel locations from original spliced image.

In order to reduce the computational complexity of this design, we plan to make decision only on edge pixels. Edge pixels include spliced region edges and authentic edges from both spliced region and authentic region. Since we are aiming to localize spliced region, only spliced region boundary should be considered as positive samples in our classification model. Furthermore, instead of using binary labels as edge pixels ground truth, we plan to use continuous label between -1 to 1. Feature extracted between neighborhood pixels will be very similar to each other. If we use totally different label for adjacent pixels, classifier may be too confused to give good prediction. In this case, the problem will be altered to a regression problem. Multiple regression model will be trained to better adapt to different image content. Then, we will apply post-processing on the binary edge map and finally output spliced region masks.

This preliminary design still needs careful analysis and modifications. For example, since we are making decision on edge pixels only, the coarseness of our output mask is hugely dependent on the edge detection performance at the very beginning. Also, in post-processing step, there inevitably exist edge pixels that are mis-classified. When we are filling (dyeing) the spliced contour to generate spliced region mask, those pixels may result in too much false alarms, thus dragging the performance of current method. Addressing aforementioned issues will be our main focus of this work in the future.

5.2.2 Image Forgery Detection

Different from image splicing localization, image forgery detection is a yes or no question for answering whether forgery exists in image. It seems to be more accessible than image splicing localization. However, other than splicing, there may be more than one kind of image forgery exist in the manipulated image. Figure 5.4 presents examples of three kinds

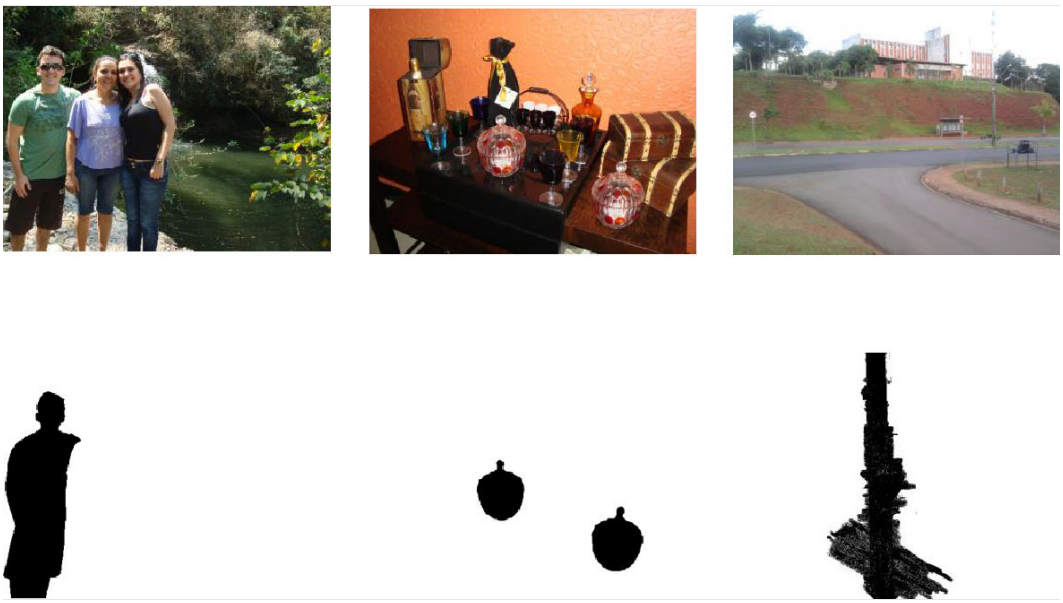


Figure 5.4: Examples of image forgeries carried out using conventional media editing tools. Images come from the dataset of the first IEEE Image Forensics Challenge organized in 2013. From left to right: splicing (alien material has been inserted in the image), copy-move (an object has been cloned), inpainting (an object has been hidden by background patches).

of image forgeries carried out using conventional media editing tools. From left to right, they are: splicing, copy-move and inpainting.

Copy-Move forgery Copy-move forgery is one of the most commonly performed manipulations on digital images. In copy-move forgery, a region from the image is copied and pasted to another region in the same image. Copy-move forgery is performed in order to hide an existing object in the image, to create a duplicate of the object or to change the meaning of the image completely. Copy-move forgery can be easy to perform but it is difficult to detect. First, the duplicated regions often share similar visual characteristics, such as texture, color, and lighting conditions, making it difficult to visually detect the tampered regions. Second, Copy-Move forgery can be achieved using basic image editing tools, such as copy-paste or cloning tools, which are widely available and easy to use. The simplicity of these techniques makes it accessible for individuals with minimal technical expertise to carry out such forgeries. Third, Copy-Move forgery operates at a local level within an

image, typically involving a small region or object. This localized nature makes it challenging to detect the tampered regions, especially when they are seamlessly blended into the surrounding content. Unlike some other types of image forgery, copy-move manipulations may not leave distinct artifacts or traces that can be easily detected by traditional forensic techniques. The forgery often involves copying and pasting regions without introducing noticeable inconsistencies in noise patterns, sharpness, or compression artifacts. Lastly, existing post-processing tools make it harder to detect copy-move forgeries. Perpetrators of copy-move forgery may employ various post-processing techniques to further conceal their manipulations. This can include applying blurring, noise addition, or color adjustments to the tampered regions, making it harder to detect the duplicated content.

In terms of the detection methods of copy-move forgery, neural networks have proved its potential. Siamese Networks can identify duplicated regions by comparing patches within the same image. This approach is effective in detecting copy-move forgeries even when the duplicated regions are subjected to geometric transformations or slight modifications. Deep Attention Models are able to allocate attention to distinctive and discriminative parts of the image, making them robust to varying lighting conditions, background clutter, and occlusions. Attention mechanisms enable the model to effectively locate and classify manipulated regions. Capsule Networks have shown promise in copy-move forgery detection by capturing hierarchical relationships between image elements. These models can recognize object instances and spatial arrangements, making them well-suited for detecting duplicated regions in copy-move forgeries. These models have shown good performance in copy-move forgery detection, but they may not be able to tackle the combination of forgeries of splicing, copy-move and inpainting. Also, these models are too large to be considered in actual products. It shows a promising starting point to for us to adopt green learning methodology.

Inpainting Image inpainting was initially used for reconstructing the deteriorated portions of the image by considering neighbouring areas of the distorted regions. But the

forgers are trying their finest to attempt a kind of forgery such that it seems real. Using image inpainting to perform manipulations on the image is one such technique. The process of inpainting involves analyzing the surrounding information and utilizing it to estimate the missing pixels. Both traditional and deep learning based method have been proved to achieve plausible performance.

Traditionally, people use Exemplar-based methods to do inpainting. These methods rely on finding similar patches or regions in the image and copying them to fill in the missing areas. Examples include the PatchMatch algorithm and exemplar-based texture synthesis. Other than Exemplar-based methods, partial differential equation (PDE)-based methods is also proved successful. These methods use mathematical equations to propagate information from the known regions to the unknown areas. The Navier-Stokes equation and the heat equation are commonly used for inpainting. The inpainting algorithm proposed by Bertalmio et al. is a well-known PDE-based approach.

With the advancements in deep learning, neural network-based approaches have shown remarkable performance in inpainting tasks. These methods leverage large-scale training datasets to learn the underlying structures and context in images, enabling them to generate visually plausible and realistic inpainted results. Some state-of-the-art deep learning-based inpainting methods include Context Encoder (CE), GANs, EdgeConnect and Generative Query Network. CE is an autoencoder-based network that is trained to inpaint missing regions conditioned on the surrounding context. GANs utilize adversarial training framework helps in generating more realistic and visually convincing inpainting results. Notable GAN-based inpainting models include DeepFill, Global and Local Consistent Image Completion (GLCIC), and Partial Convolutional Neural Network (PCNN). EdgeConnect focuses on maintaining the structure and edges of the missing regions. It utilizes an edge generation network and an inpainting network to generate high-quality inpainted images with coherent edges. Generative Query Network (GQN) is a more recent inpainting approach that

leverages a neural network to learn the representation of scenes and their underlying 3D structure. It can inpaint missing parts of images by predicting the plausible content based on the learned scene representation.

There are many successful models to do inpainting. However, for inpainting forgery detection, it is often merged in image forgery detection together with copy-move detection and image splicing localization. It shows a promising starting point to for us to adopt green learning methodology in combined or multiple image forgery detection.

References

[1] Irene Amerini, Rudy Becarelli, Roberto Caldelli, and Andrea Del Mastio. Splicing forgeries localization through the use of first digit features. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 143–148. IEEE, 2014.

[2] Zohreh Azizi and C-C Jay Kuo. Pager: Progressive attribute-guided extendable robust image generation. *arXiv preprint arXiv:2206.00162*, 2022.

[3] Patrick Bas, Tomáš Filler, and Tomáš Pevnỳ. " break our steganographic system": the ins and outs of organizing boss. In *International workshop on information hiding*, pages 59–70. Springer, 2011.

[4] Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2444–2447. IEEE, 2011.

[5] Tiziano Bianchi and Alessandro Piva. Detection of nonaligned double jpeg compression based on integer periodicity maps. *IEEE transactions on Information Forensics and Security*, 7(2):842–848, 2011.

[6] Tiziano Bianchi and Alessandro Piva. Image forgery localization via block-grained analysis of jpeg artifacts. *IEEE Transactions on Information Forensics and Security*, 7(3):1003–1017, 2012.

[7] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181–1193, 2018.

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[9] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3291–3300, 2018.

[10] Hong-Shuo Chen, Mozhdeh Rouhsedaghat, Hamza Ghani, Shuowen Hu, Suya You, and C-C Jay Kuo. Defakehop: A light-weight high-performance deepfake detector. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[11] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukás. Determining image origin and integrity using sensor noise. *IEEE Transactions on information forensics and security*, 3(1):74–90, 2008.

[12] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017.

[13] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[14] Yueru Chen and C-C Jay Kuo. Pixelhop: A successive subspace learning (ssl) method for object recognition. *Journal of Visual Communication and Image Representation*, page 102749, 2020.

[15] Yueru Chen, Mozhdeh Rouhsedaghat, Suya You, Raghuveer Rao, and C-C Jay Kuo. Pixelhop++: A small successive-subspace-learning-based (ssl-based) model for image classification. *arXiv preprint arXiv:2002.03141*, 2020.

[16] Yueru Chen, Zhuwei Xu, Shanshan Cai, Yujian Lang, and C-C Jay Kuo. A saak transform approach to efficient, scalable and robust handwritten digits recognition. In *2018 Picture Coding Symposium (PCS)*, pages 174–178. IEEE, 2018.

[17] Giovanni Chierchia, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. A bayesian-mrf approach for prnu-based image forgery detection. *IEEE Transactions on Information Forensics and Security*, 9(4):554–567, 2014.

[18] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul 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*, pages 8789–8797, 2018.

[19] Remi Cogranne and Florent Retraint. An asymptotically uniformly most powerful test for lsb matching detection. *IEEE transactions on information forensics and security*, 8(3):464–476, 2013.

[20] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2015.

[21] Davide Cozzolino, Giovanni Poggi, and Luisa 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*, pages 159–164, 2017.

[22] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11065–11074, 2019.

[23] Ahmet Emir Dirik and Nasir Memon. Image tamper detection based on demosaicing artifacts. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1497–1500. IEEE, 2009.

[24] Hany Farid. Exposing digital forgeries from jpeg ghosts. *IEEE transactions on information forensics and security*, 4(1):154–160, 2009.

[25] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Image forgery localization via fine-grained analysis of cfa artifacts. *IEEE Transactions on Information Forensics and Security*, 7(5):1566–1577, 2012.

[26] Tomáš Filler and Jessica Fridrich. Gibbs construction in steganography. *IEEE Transactions on Information Forensics and Security*, 5(4):705–720, 2010.

[27] Tomáš Filler, Jan Judas, and Jessica Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3):920–935, 2011.

[28] Jessica Fridrich and Miroslav Goljan. On estimation of secret message length in lsb steganography in spatial domain. In *Security, steganography, and watermarking of multimedia contents VI*, volume 5306, pages 23–34. SPIE, 2004.

[29] Jessica Fridrich, Miroslav Goljan, and Rui Du. Detecting lsb steganography in color, and gray-scale images. *IEEE multimedia*, 8(4):22–28, 2001.

[30] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on information Forensics and Security*, 7(3):868–882, 2012.

[31] Hongyu Fu, Yijing Yang, Vinod K Mishra, and C-C Jay Kuo. Classification via subspace learning machine (slm): Methodology and performance evaluation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[32] Xiou Ge, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Core: A knowledge graph entity type prediction method via complex space regression and embedding. *Pattern Recognition Letters*, 2022.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[34] Vojtěch Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *2012 IEEE International workshop on information forensics and security (WIFS)*, pages 234–239. IEEE, 2012.

[35] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1):1–13, 2014.

[36] Donghui Hu, Liang Wang, Wenjie Jiang, Shuli Zheng, and Bin Li. A novel image steganography method via deep convolutional generative adversarial networks. *IEEE Access*, 6:38303–38314, 2018.

[37] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[38] Pranav Kadam, Min Zhang, Shan Liu, and C-C Jay Kuo. Unsupervised point cloud registration via salient points analysis (spa). In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 5–8. IEEE, 2020.

[39] Pranav Kadam, Min Zhang, Shan Liu, and C-C Jay Kuo. Gpco: An unsupervised green point cloud odometry method. *arXiv preprint arXiv:2112.04054*, 2021.

[40] Pranav Kadam, Min Zhang, Shan Liu, and C-C Jay Kuo. R-pointhop: A green, accurate and unsupervised point cloud registration method. *arXiv preprint arXiv:2103.08129*, 2021.

[41] Kadam, Pranav and Zhang, Min and Liu, Shan and Kuo, C-C Jay. R-pointhop: A green, accurate, and unsupervised point cloud registration method. *IEEE Transactions on Image Processing*, 2022.

[42] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[43] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[44] Andrew D Ker. Steganalysis of lsb matching in grayscale images. *IEEE signal processing letters*, 12(6):441–444, 2005.

[45] Andrew D Ker and Rainer Böhme. Revisiting weighted stego-image steganalysis. In *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 56–72. SPIE, 2008.

[46] C-C Jay Kuo. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41:406–413, 2016.

[47] C-C Jay Kuo. The cnn as a guided multilayer recos transform [lecture notes]. *IEEE signal processing magazine*, 34(3):81–89, 2017.

[48] C-C Jay Kuo and Yueru Chen. On data-driven saak transform. *Journal of Visual Communication and Image Representation*, 50:237–246, 2018.

[49] C-C Jay Kuo and Azad M Madni. Green learning: Introduction, examples and outlook. *Journal of Visual Communication and Image Representation*, page 103685, 2022.

[50] C-C Jay Kuo, Min Zhang, Siyang Li, Jiali Duan, and Yueru Chen. Interpretable convolutional neural networks via feedforward design. *Journal of Visual Communication and Image Representation*, 2019.

[51] Xuejing Lei, Wei Wang, and C-C Jay Kuo. Genhop: An image generation method based on successive subspace learning. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3314–3318. IEEE, 2022.

[52] Xuejing Lei, Ganning Zhao, and C-C Jay Kuo. Nites: A non-parametric interpretable texture synthesis method. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1698–1706. IEEE, 2020.

[53] Xuejing Lei, Ganning Zhao, Kaitai Zhang, and C-C Jay Kuo. Tghop: An explainable, efficient and lightweight method for texture generation. *arXiv preprint arXiv:2107.04020*, 2021.

[54] Bin Li, Ming Wang, Jiwu Huang, and Xiaolong Li. A new cost function for spatial image steganography. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4206–4210. IEEE, 2014.

[55] Chang-Tsun Li and Yue Li. Color-decoupled photo response non-uniformity for digital image forensics. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(2):260–271, 2011.

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

[57] Ke Li, Tianhao Zhang, and Jitendra Malik. Diverse image synthesis from semantic layouts via conditional imle. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4220–4229, 2019.

[58] Weihai Li, Yuan Yuan, and Nenghai Yu. Passive detection of doctored jpeg image via block artifact grid extraction. *Signal Processing*, 89(9):1821–1829, 2009.

[59] Xiaolong Li, Tieyong Zeng, and Bin Yang. Detecting lsb matching by applying calibration technique for difference image. In *Proceedings of the 10th ACM workshop on Multimedia and security*, pages 133–138, 2008.

[60] Zhouchen Lin, Junfeng He, Xiaoou Tang, and Chi-Keung Tang. Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis. *Pattern Recognition*, 42(11):2492–2501, 2009.

[61] Xiaofeng Liu, Fangxu Xing, Chao Yang, C-C Jay Kuo, Suma Babu, Georges El Fakhri, Thomas Jenkins, and Jonghye Woo. Voxelhop: Successive subspace learning for als disease classification using structural mri. *arXiv preprint arXiv:2101.05131*, 2021.

[62] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[63] Ji-cang Lu, Fen-lin Liu, and Xiang-yang Luo. Selection of image features for steganalysis based on the fisher criterion. *Digital investigation*, 11(1):57–66, 2014.

[64] Weiqi Luo, Zhenhua Qu, Jiwu Huang, and Guoping Qiu. A novel method for detecting cropped and recompressed image block. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 2, pages II–217. IEEE, 2007.

[65] Siwei Lyu, Xunyu Pan, and Xing Zhang. Exposing region splicing forgeries with blind local noise estimation. *International journal of computer vision*, 110(2):202–221, 2014.

[66] Vasileios Magoulianitis, Yijing Yang, and C-C Jay Kuo. Hunis: High-performance unsupervised nuclei instance segmentation. *arXiv preprint arXiv:2203.14887*, 2022.

[67] Babak Mahdian and Stanislav Saic. Using noise inconsistencies for blind image forensics. *Image and Vision Computing*, 27(10):1497–1503, 2009.

[68] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of gan-generated fake images over social networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 384–389. IEEE, 2018.

[69] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 83–92. IEEE, 2019.

[70] Masoud Monajatipoor, Mozhdeh Rouhsedaghat, Liunian Harold Li, Aichi Chien, C-C Jay Kuo, Fabien Scalzo, and Kai-Wei Chang. Berthop: An effective vision-and-language model for chest x-ray disease diagnosis. *arXiv preprint arXiv:2108.04938*, 2021.

[71] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019(5):532–1, 2019.

[72] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!*, pages 1–1. 2019.

[73] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[74] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks. In *Media Watermarking, Security, and Forensics 2015*, volume 9409, pages 171–180. SPIE, 2015.

[75] Tabares-Soto Reinel, Arteaga-Arteaga Harold Brayan, Bravo-Ortiz Mario Alejandro, Mora-Rubio Alejandro, Arias-Garzon Daniel, Alzate-Grisales Jesus Alejandro, Burbano-Jacome Alejandro Buenaventura, Orozco-Arias Simon, Isaza Gustavo, and Ramos-Pollan Raul. Gbras-net: a convolutional neural network architecture for spatial image steganalysis. *IEEE Access*, 9:14340–14350, 2021.

[76] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2019.

[77] Mozhdeh Rouhsedaghat, Masoud Monajatipoor, Zohreh Azizi, and C-C Jay Kuo. Successive subspace learning: An overview. *arXiv preprint arXiv:2103.00121*, 2021.

[78] Mozhdeh Rouhsedaghat, Yifan Wang, Xiou Ge, Shuowen Hu, Suya You, and C-C Jay Kuo. Facehop: A light-weight low-resolution face gender classification method. *arXiv preprint arXiv:2007.09510*, 2020.

[79] Mozhdeh Rouhsedaghat, Yifan Wang, Shuowen Hu, Suya You, and C-C Jay Kuo. Low-resolution face recognition in resource-constrained environments. *Pattern Recognition Letters*, 149:193–199, 2021.

[80] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. Image splicing localization using a multi-task fully convolutional network (mfcn). *Journal of Visual Communication and Image Representation*, 51:201–209, 2018.

[81] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. Ssgan: secure steganography based on generative adversarial networks. In *Pacific Rim Conference on Multimedia*, pages 534–544. Springer, 2017.

[82] Weixuan Tang, Bin Li, Shunquan Tan, Mauro Barni, and Jiwu Huang. Cnn-based adversarial embedding for image steganography. *IEEE Transactions on Information Forensics and Security*, 14(8):2074–2087, 2019.

[83] Weixuan Tang, Haodong Li, Weiqi Luo, and Jiwu Huang. Adaptive steganalysis based on embedding probabilities of pixels. *IEEE Transactions on Information Forensics and Security*, 11(4):734–745, 2015.

[84] Weixuan Tang, Shunquan Tan, Bin Li, and Jiwu Huang. Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10):1547–1551, 2017.

[85] Bin Wang, Guangtao Wang, Jing Huang, Jiaxuan You, Jure Leskovec, and C-C Jay Kuo. Inductive learning on commonsense knowledge graph completion. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[86] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8695–8704, 2020.

[87] Yifan Wang, Zhanxuan Mei, Ioannis Katsavounidis, and C-C Jay Kuo. Dcst: a data-driven color/spatial transform-based image coding method. In *Applications of Digital Image Processing XLIV*, volume 11842, page 118420P. International Society for Optics and Photonics, 2021.

[88] Yun-Cheng Wang, Xiou Ge, Bin Wang, and C-C Jay Kuo. Kgboost: A classification-based knowledge base completion method with negative sampling. *Pattern Recognition Letters*, 2022.

[89] Songtao Wu, Shenghua Zhong, and Yan Liu. Deep residual learning for image steganalysis. *Multimedia tools and applications*, 77(9):10437–10453, 2018.

[90] Tian Xie, Bin Wang, and C-C Jay Kuo. Graphhop: An enhanced label propagation method for node classification. *arXiv preprint arXiv:2101.02326*, 2021.

[91] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. Structural design of convolutional neural networks for steganalysis. *IEEE Signal Processing Letters*, 23(5):708–712, 2016.

[92] Jianhua Yang, Danyang Ruan, Jiwu Huang, Xiangui Kang, and Yun-Qing Shi. An embedding cost learning framework using gan. *IEEE Transactions on Information Forensics and Security*, 15:839–851, 2019.

[93] Yijing Yang, Wei Wang, Hongyu Fu, and C-C Jay Kuo. On supervised feature selection from high dimensional feature spaces. *arXiv preprint arXiv:2203.11924*, 2022.

[94] Jian Ye, Jiangqun Ni, and Yang Yi. Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557, 2017.

[95] Shuiming Ye, Qibin Sun, and Ee-Chien Chang. Detecting digital image forgeries by measuring inconsistencies of blocking artifact. In *2007 IEEE International Conference on Multimedia and Expo*, pages 12–15. Ieee, 2007.

[96] Mehdi Yedroudj, Frédéric Comby, and Marc Chaumont. Yedroudj-net: An efficient cnn for spatial steganalysis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2092–2096. IEEE, 2018.

[97] Weike You, Hong Zhang, and Xianfeng Zhao. A siamese cnn for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 16:291–306, 2020.

[98] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7556–7566, 2019.

[99] Kaitai Zhang, Hong-Shuo Chen, Ye Wang, Xiangyang Ji, and C-C Jay Kuo. Texture analysis via hierarchical spatial-spectral correlation (hssc). In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4419–4423. IEEE, 2019.

[100] Kaitai Zhang, Hong-Shuo Chen, Xinfeng Zhang, Ye Wang, and C-C Jay Kuo. A data-centric approach to unsupervised texture segmentation using principle representative patterns. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1912–1916. IEEE, 2019.

[101] Kaitai Zhang, Bin Wang, Wei Wang, Fahad Sohrab, Moncef Gabbouj, and C-C Jay Kuo. Anomalyhop: An ssl-based image anomaly localization method. *arXiv preprint arXiv:2105.03797*, 2021.

[102] Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. Steganogan: high capacity image steganography with gans. *arXiv preprint arXiv:1901.03892*, 2019.

[103] Min Zhang, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Unsupervised feedforward feature (uff) learning for point cloud classification and segmentation. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 144–147. IEEE, 2020.

[104] Min Zhang, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Gsip: Green semantic segmentation of large-scale indoor point clouds. *arXiv preprint arXiv:2109.11835*, 2021.

[105] Min Zhang, Yifan Wang, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Pointhop++: A lightweight learning model on point sets for 3d classification. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3319–3323. IEEE, 2020.

[106] Min Zhang, Haoxuan You, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Pointhop: An explainable machine learning method for point cloud classification. *IEEE Transactions on Multimedia*, 2020.

[107] Ru Zhang, Feng Zhu, Jianyi Liu, and Gongshen Liu. Depth-wise separable convolutions and multi-level pooling for an efficient spatial cnn-based steganalysis. *IEEE Transactions on Information Forensics and Security*, 15:1138–1150, 2019.

[108] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

[109] Zhiruo Zhou, Hongyu Fu, Suya You, Christoph C Borel-Donohue, and C-C Jay Kuo. Uhp-sot: An unsupervised high-performance single object tracker. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–5. IEEE, 2021.

[110] Zhiruo Zhou, Hongyu Fu, Suya You, and C-C Jay Kuo. Unsupervised lightweight single object tracking with uhp-sot++. *arXiv preprint arXiv:2111.07548*, 2021.

[111] Zhiruo Zhou, Hongyu Fu, Suya You, and C-C Jay Kuo. Gusot: Green and unsupervised single object tracking for long video sequences. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2022.

[112] Zhiruo Zhou, Hongyu Fu, Suya You, C-C Jay Kuo, et al. Uhp-sot++: An unsupervised lightweight single object tracker. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[113] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.

[114] Yao Zhu, Saksham Suri, Pranav Kulkarni, Yueru Chen, Jiali Duan, and C-C Jay Kuo. An interpretable generative model for handwritten digits synthesis. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1910–1914. IEEE, 2019.

[115] Yao Zhu, Xinyu Wang, Hong-Shuo Chen, Ronald Salloum, and C-C Jay Kuo. A-pixelhop: A green, robust and explainable fake-image detector. *arXiv preprint arXiv:2111.04012*, 2021.