DEEP LEARNING FOR IMAGE STEGANOGRAPHY AND STEGANALYSIS CHALLENGES, ADVANCES, AND OPPORTUNITIES

BY

YASSINE YOUSFI

MS, Ecole Centrale De Lille, 2018

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate School of Binghamton University
State University of New York
2022

Accepted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate School of Binghamton University
State University of New York
2022
May 10^{th}, 2022

Jessica Fridrich, Chair and Faculty Advisor
Department of Electrical and Computer Engineering, Binghamton University

Emrah Akyol, Member
Department of Electrical and Computer Engineering, Binghamton University

Xiaohua (Edward) Li, Member
Department of Electrical and Computer Engineering, Binghamton University

Timothy J. Singler, Outside Examiner
Department of Mechanical Engineering, Binghamton University

Abstract

Deep learning has proven incredibly successful in a plethora of fields. In computer vision, deep neural networks are now the state-of-the-art for a variety of tasks. At the first glance, steganography and steganalysis appear to be very much different tasks than classical computer vision tasks, yet deep learning, especially convolutional neural networks, popularized by the computer vision field, have outperformed all classical feature-based approaches for detecting steganography.

Intuitively, steganographic embedding changes are weak, noise-like signals executed primarily in complex content, such as textures and edges. Since computer vision classifies and categorizes content, it is also suitable for detecting the presence of noise-like stego signals modulated by content.

In this dissertation, we focus on refactoring steganography detectors with more modern and general components, qualitatively understanding their strengths and failure cases, and using them algorithmically to improve steganography. First, we show that many custom ingredients long believed to be necessary for successfully training a deep neural network for steganography detection can be omitted in favor of more general-purpose convolutional architectures with very few domain-specific changes. Next, we focus on understanding what makes deep neural networks superior to their classical feature-based predecessors. Lastly, we use these powerful steganography detectors as a feedback loop in novel batch steganography algorithms, which allocate more payload in images where state-of-the-art detectors fail to detect steganography.

Acknowledgements

I would like to thank my Ph.D. advisor, Jessica Fridrich, for her guidance and trust at every stage of my Ph.D. studies. I have cherished every time spent together in the lab, and our long discussions about research, coffee, and tree-climbing goats.

I would also like to thank the members of Digital Data Embedding laboratory and the visiting students and scholars. Their help, collaboration, and friendship were greatly appreciated.

In addition, I would like to express my appreciation to my Ph.D. committee members and outside examiner, Emrah Akyol, Xiaohua (Edward) Li, and Timothy J. Singler for their time and helpful advice.

Finally, I would like to thank my family, Rachel Weisbrot, and Ouwaïs Marrakchi Benjaafar for their love and unconditional support.

Yassine Yousfi

Preface

Deep Learning is ubiquitous in steganography and steganalysis. Convolutional Neural Networks (CNNs), in particular, clearly outcompete the previous generation of steganography detectors, the so-called Rich Models. During his four years of Ph.D. studies at Binghamton University, the author's research focused on advancing state-of-the-art of steganography and steganalysis using Deep Learning models and understanding their strengths and weaknesses. This dissertation is written as the final requirement of the author's Ph.D. degree and describes in detail selected author's peer-reviewed and published contributions.

In Chapter 1, we introduce the fundamental concepts in steganography and steganalysis. We describe the evolution of Machine Learning and Deep Learning techniques from the early feature-based steganalysis to the more recent CNN architectures trained in an end-to-end fashion using large datasets with minimal domain-specific elements.

While believed to universally outperform Rich Models, CNNs were shown to struggle in some settings of JPEG steganalysis. Chapter 2 describes this intriguing failure and proposes a practical solution based on the so-called OneHotConv layer.

Chapter 3 also challenges the common belief that CNNs for JPEG steganalysis should be trained separately for each Quality Factor. It shows that similar performance can be obtained by training CNNs on an entire range of JPEG qualities, which drastically cuts down the complexity of building practical JPEG steganography detectors.

The ALASKA II steganalysis challenge hosted by Kaggle saw the rise of a new type of detectors, computer-vision CNNs pretrained on ImageNet and refined for steganalysis. Chapter 4 describes these detectors, which significantly outperformed the state-of-the-art network for steganalysis, the SRNet. These detectors were trained without the so-called Pair Constraint, also believed to be a crucial element for training CNNs for steganalysis. They were also trained on an even larger range of JPEG qualities than prescribed in Chapter 3.

Chapter 5 shows that ImageNet pretrained models can gain even more from targeted modifications of their architectures: disabling strides in the first layer and inserting a number of unpooled and

unstrided convolutions in their early layers. The modified architectures outperform the state-of-the-art in steganalysis on a range of datasets and steganographic algorithms.

In Chapter 6, we ask the question "What makes these CNNs significantly outperform the Rich Models?" We show that unlike Rich Models which operate on macroscopic quantities of local statistics, CNNs can leverage highly localized artifacts left by steganography.

With these great steganography detectors in hand and a better understanding of their strengths and weaknesses, Chapter 7 proposes new detector-informed batch steganography algorithms, which use feedback from state-of-the-art detectors to spread the secret payload across multiple cover images. Chapter 7 also pays close attention to the impact of the information available to the Warden and to her pooling strategies for a more comprehensive assessment of security.

The dissertation is concluded in Chapter 8.

Contents

List of tables

List of figures

Chapter 1

Introduction and preliminaries

Steganography is the art of covert communication when secrets messages are communicated in ordinary looking cover objects. The goal is to make steganographic communication indistinguishable from regular exchange of information during which no secrets are passed between communicating parties. To do so, the sender hides the message in a cover object while taking good care of keeping it innocuous. Digital media, such as images, are particularly suitable cover objects because of their ubiquity and because they can be slightly modified without changing their appearance, potentially thus able to hold large messages.

The task of detecting the presence of steganography is called steganalysis. In the case of image steganography, it is performed by inspecting statistical properties of the intercepted images and comparing them to statistical properties of innocent cover images. Image steganalysis is a rather difficult task because natural images contain many indeterministic components due to the acquisition conditions and to the high diversity and complexity introduced during development from the raw capture, post-processing, editing, saving, and even sharing.

1.1 Steganographic channel

There are three main approaches for covert communication: steganography by cover modification, cover synthesis, and cover selection [3]. This dissertation will only consider steganography by cover modification.

Steganography is often described using the prisoners' problem. Alice and Bob are allowed to com-

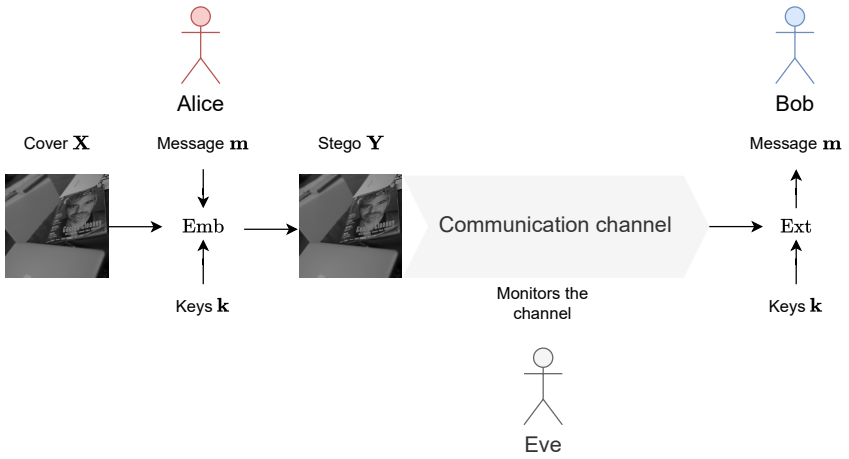


Figure 1.1.1: Steganographic channel.

municate but all messages they exchange are closely monitored by warden Eve looking for data potentially hidden in their communication [4]. Once Eve detects the mere presence of steganography, the channel is considered compromised.

We also adopt the Kerckhoffs' principle, which assumes that Eve knows everything about the steganographic channel apart from the secret key. This includes the hiding algorithm, the cover source (but not the cover image used by Alice), the message source and size (but not the exact message). The Kerckhoffs' principle is often used as a worst case scenario to measure steganographic security.

A steganographic system consists of a cover source $\{\mathcal{C}, P^{(c)}\}$, a message source $\{\mathcal{M}, P^{(m)}\}$, a key source $\{\mathcal{K}, P^{(k)}\}$, and embedding and extracting functions Emb and Ext. The pair $\{\Omega, P^{(\omega)}\}$ denotes the set of all possible objects in Ω equipped with a probability distribution $P^{(\omega)}$. The embedding function $\text{Emb} : \mathcal{C} \times \mathcal{M} \times \mathcal{K} \to \mathcal{Y}$ takes a cover object, a message Alice wants to communicate, a shared secret key and creates a stego object carrying the message $\mathbf{y} = \text{Emb}(\mathbf{x}, \mathbf{m}, \mathbf{k}) \in \{\mathcal{Y}, P^{(y)}\}$. The extraction function $\text{Ext} : \mathcal{Y} \times \mathcal{K} \to \mathcal{M}$ takes the stego object and secret key to extract the secret $\mathbf{m} = \text{Ext}(\text{Emb}(\mathbf{x}, \mathbf{m}, \mathbf{k}), \mathbf{k})$. Note that when using Ext, Bob does not need the cover object $\mathbf{x}$ to extract the secret message from $\mathbf{y}$. The steganographic channel is visualized in Figure 1.1.1.

1.2 Image steganography

Most early steganographic systems using digital images were hiding messages in the Least Significant Bits (LSBs) of the image's pixel values. In fact a survey of tools capable of hiding data in digital images available on the internet found that 1024 out of 2863 (36%) were embedding messages by

Figure 1.2.1: Cover (top) and stego (bottom) images and their LSB plane.

manipulating LSBs [5]. The most primitive way to embed a message in an image is by replacing the LSBs of the image pixels to match the message in a bitstream form. Alice can also randomly select which pixel locations to choose from using a secret key $\mathbf{k}_{\text{loc}}$ shared with Bob.

LSB Replacement (LSBR) is unfortunately very detectable [6; 7; 8; 9; 10]. Another algorithm based on LSBs is LSB Matching (LSBM), which matches the cover LSB to the message bit by randomly changing the pixel value by $+1$ or -1. LSBM is generally more secure than LSBR but still very detectable [11; 12; 13]. It is used as the basis for more secure content adaptive algorithms.

The probability of change of pixel i is called the change rate β_i. For a naive implementation of LSBR, embedding a message of length m bits in an image consisting of n pixels, the expected value of the change rate is $\beta_i = \frac{m}{2n}$ ($\frac{m}{n}$ pixels are changed on average with probability $\frac{1}{2}$). Modern steganographic algorithms make use of coding to improve the embedding efficiency (bits communicated per embedding changes). The so-called Syndrome Trellis Codes (STCs) [14] are widely used in steganographic

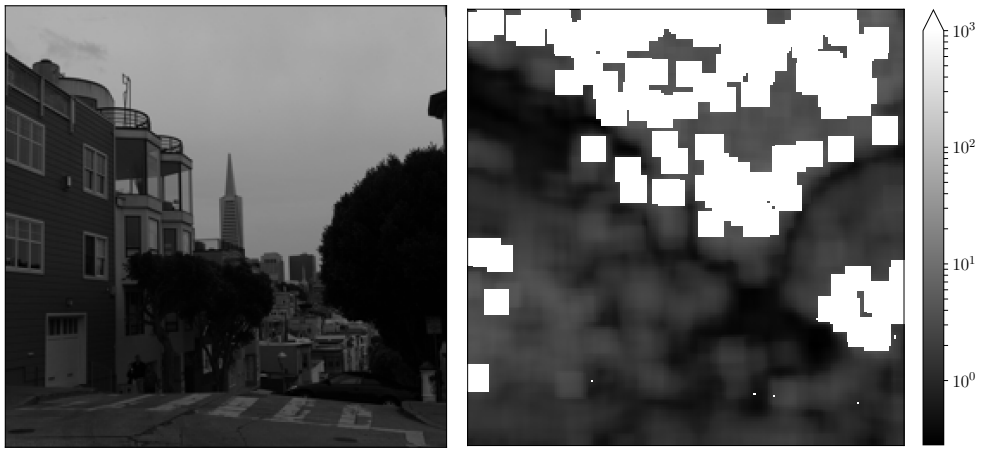


Figure 1.2.2: HILL costs of an image. Notice the square regions with very large costs, the so called wet costs, which prohibit embedding changes around very smooth regions of an image.

implementations because they achieve near optimal efficiency in terms of the Rate-Distortion (RD) bound

$$\beta \geq H_3^{-1}(\alpha), \tag{1.2.1}$$

where H_3^{-1} is the inverse of the ternary entropy function, which we assume here for simplicity, assigning the same probability of change β_i for $+1$ or -1.

$$H_3(\beta_i) = -(1-2\beta_i)\log_2(1-2\beta_i) - 2\beta_i \log_2 \beta_i. \tag{1.2.2}$$

Steganographic simulators generally assume optimal coding by operating on the RD bound, i.e.

$$\sum_{i=1}^{n} H_3(\beta_i) = n\alpha. \tag{1.2.3}$$

The most popular content-adaptive strategies use heuristically defined costs ρ_i of changing the i-th cover element. The cost assignment is typically designed experimentally and is usually the central element of each embedding algorithm. An example of steganographic costs is given in Figure 1.2.2.

The total cost of all executed embedding changes is defined as the distortion

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \rho_i [x_i \neq y_i]. \tag{1.2.4}$$

Decomposing the distortion as a sum of costs of individual changes means that those changes do not

interact with each other. Embedding changes that interact were proposed in [15; 16; 17]. Also note that the cost of no change ($x_i = y_i$) is zero. In practice, cost based schemes determine the optimal change rates by solving

$$\begin{aligned} \min E[D(\mathbf{x}, \mathbf{y})] &= \sum_{i=1}^{n} \rho_i \beta_i \\ \text{st} \sum_{i=1}^{n} H_3(\beta_i) &= n\alpha. \end{aligned} \tag{1.2.5}$$

The solution is

$$\beta_i = \frac{e^{-\lambda \rho_i}}{1 + 2e^{-\lambda \rho_i}}, \tag{1.2.6}$$

where λ is a Lagrange multiplier found using a binary search.

The most popular and secure algorithms include WOW [18], S-UNIWARD [19], and HILL [20] in the spatial domain and J-UNIWARD [19], UED [21], and UERD [22] in the JPEG domain. Figure 1.2.2 shows HILL costs of an image.

Another approach to steganography called model based steganography [23; 24; 25; 26; 27; 28] imposes a statistical model on the cover elements $p_i^{(c)}$ and minimizes the effect of steganography on the model measured using Kullback–Leibler divergence:

$$\begin{aligned} \min D_{\text{KL}}(p^{(c)} || p^{(y)}) &= \sum_{i} D_{\text{KL}}(p_i^{(c)} || p_i^{(y)}(\beta_i)) \\ \text{st} \sum_{i=1}^{n} H_3(\beta_i) &= n\alpha. \end{aligned} \tag{1.2.7}$$

A tangential body of work considers the batch steganography problem [29; 30; 31; 32; 33; 34; 35], which consists of sending a bag of B images instead of a single image. Batch steganography gives an opportunity to spread the payload in a more adaptive fashion, i.e. embed more payload in textured and complex images, and less payload in smooth images.

1.3 Image Steganalysis

In the most general way, the steganalysis problem Eve faces can then be formulated as a binary hypothesis testing

$$\mathrm{H}_0 : \mathbf{x} \sim P^{(c)}, \tag{1.3.1}$$

$$\mathrm{H}_1 : \mathbf{x} \sim P^{(y)}. \tag{1.3.2}$$

However, Eve does not know $P^{(c)}$ nor $P^{(y)}$ since covers and stegos live in the complicated high dimensional space of natural images. In practice, Eve's detector is a binary mapping $f : \mathcal{X} \cup \mathcal{Y} \to \{0, 1\}$ where 1 is predicting that the intercepted image is a stego image. Eve estimates (or learns) f using a data driven approach which will be explained in more details in 1.4.

Note that in some special cases, equivalents of $P^{(c)}$ and $P^{(y)}$ can be derived or approximated [36; 37; 38; 11; 39; 40; 41], in which case Eve's best detector is a Likelihood Ratio Test (LRT) following the Neyman–Pearson lemma.

In this dissertation, we will measure the performance of Eve's detectors using the following scalar descriptors:

- The minimum probability of error under equal priors.

$$P_{\mathrm{E}} = \frac{1}{2} \min(P_{\mathrm{FA}} + P_{\mathrm{MD}}), \tag{1.3.3}$$

where P_{FA} stands for the probability of False Alarm (detecting a cover image as stego) and $P_{\mathrm{MD}} = 1 - P_D$ stands for the probability of Missed Detection (detecting a stego image as cover).

- The Missed Detection rate at 5% False Alarm rate (MD5), which was proposed as a metric in the ALASKA I steganalysis challenge [42]

$$\mathrm{MD5} = P_{\mathrm{MD}}(P_{\mathrm{FA}} = 5\%). \tag{1.3.4}$$

- The False Alarm rate at 50% Missed Detection rate (FP50), which was also proposed as a metric in the ALASKA I steganalysis challenge

$$\mathrm{FP50} = P_{\mathrm{FA}}(P_{\mathrm{MD}} = 50\%). \tag{1.3.5}$$

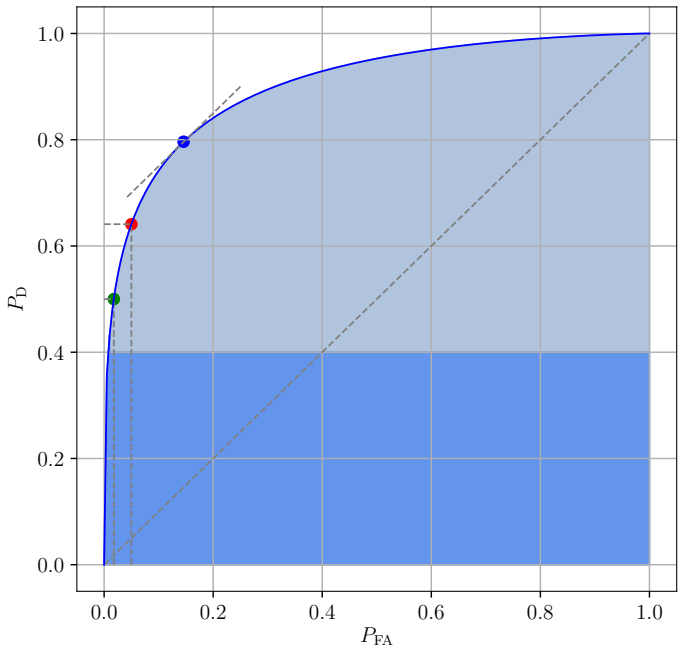


Figure 1.3.1: ROC curve of a detector and its' performance metrics. wAUC = 0.9305 which is the sum of the dark shaded blue area weighted 2× and the light shaded blue area. $P_{\mathrm{E}} = 0.1749$ which can be found using the point where a line with unit slope is tangent to the ROC curve (blue). MD5 = 0.3590 (red) and FP50 = 0.0178 (green).

- The weighted Area Under the Curve (wAUC) which is a weighted area under the Receiver Operating Characteristic (ROC) of a detector, which was proposed as a metric in the ALASKA II steganalysis challenge [43]

$$\mathrm{wAUC} = \int_0^1 w(P_{\mathrm{D}}(P_{\mathrm{FA}}))P_{\mathrm{D}}(P_{\mathrm{FA}})\mathrm{d}P_{\mathrm{FA}}, \tag{1.3.6}$$

  Where $w(P_{\mathrm{D}}) \propto 2$ if $P_{\mathrm{D}} < 0.4$ and $w(P_{\mathrm{D}}) \propto 1$ if $P_{\mathrm{D}} \geq 0.4$ with a multiplicative factor normalizing the wAUC to be in the interval $[0, 1]$.

1.4 Machine Learning in Steganography and Steganalysis

Using her Kerckhoffs power, Eve collects a finite dataset $\mathcal{D}(\mathcal{X})$ of cover objects and uses Emb and the knowledge of the payload size (usually using simulators of embedding algorithms) to build a labeled dataset of cover and stego images $\mathcal{D}(\mathcal{X}, \mathcal{Y})$. Given an error score e from those listed in 1.3,

Eve wishes to find the best detector f

$$\text{argmin}_{f \in \mathcal{F}} E[e(f)], \tag{1.4.1}$$

where the expectation is taken over all possible collected datasets $\mathcal{D}(\mathcal{X}, \mathcal{Y})$. Note that Eve specifically optimizes for a given cover and stego source $(\mathcal{X}, \mathcal{Y})$. If facing a different source, the detector found using 1.4.1 will likely underperform. This phenomenon is known as the cover/stego source mismatch [44; 45; 46; 47; 48; 49; 50].

Given that the space of all possible detectors $\mathcal{F}$ is difficult to navigate, Eve will use parametric detectors f_θ and solve the minimization problem 1.4.1 over the parameter vector θ. Moreover, early data driven approaches used fixed feature representations of images $g : \mathcal{X} \cup \mathcal{Y} \to \mathbb{R}^d$ where d is the dimensionality of the feature space. The final detector is $f_\theta \circ g$ which is the composition of the feature representation (fixed) and the classifier. Feature vectors g were heuristically crafted to increase the power of the stego signal relative to natural image content and noise.

Early machine-learning steganalyzers used separable Quadrature Mirror Filters (QMFs), Fisher Linear Discriminant analysis (FLD), and Support Vector Machine classifiers (SVM) [51; 52; 53] as classifiers. Similar techniques were also used for detection of watermarks [54]. A set of 23 features was then introduced for steganalysis in the JPEG domain [55] based on global and mode-wise histograms, co-occurrence histograms, and ad-hoc blockiness measures.

Sample transition probability matrices were proposed for JPEG steganalysis in [56; 57], which model JPEG coefficients as a Markov random process. A spatial-domain equivalent was described by Zo [58] and Pevný [59; 60]. To further improve steganalysis, features combined across different domains were proposed (the Cross-Domain Features CDF) [61].

The switch from transition probability matrices (conditional probabilities) to joint distributions (co-occurrences) began with attempts at steganlyzing HUGO [62], which aspired to preserve a very high dimensional feature formed by 3D co-occurrences. The BOSS competition spurred further development in steganalysis, which began the trend of feature "richification" that culminated in the design of rich media models [63; 64; 65; 66; 67; 68; 69] with simple classifiers [70; 71; 49; 72; 73]. Rich models were also extended to color images [74; 75; 76].

Feature sets were also proposed to target specific content-adaptive stego schemes, unlike the above cited methods which are general (blind) and do not incorporate any knowledge of the stego scheme they target besides from the collection of $D(\mathcal{X}, \mathcal{Y})$. The first targeted feature vector, the thresholded

SRM [77], was specifically designed to attack the content-adaptive scheme WOW. General selection-channel-aware features, the maxSRM and maxSRMd2 were introduced in [78] and later in [79]. SCA features use the estimated embedding change probabilities $\hat{\beta}_i$ from the analyzed image. Note that these change probabilities are generally disturbed by steganography, i.e. if Eve analyzes a cover image $\hat{\beta}_i = \beta_i$ but if Eve analyzes a stego image $\hat{\beta}_i \neq \beta_i$. SCA features for JPEG steganography were later introduced in [80] specifically for steganalysis of modern content-adaptive JPEG steganography (J-UNIWARD and UED). JPEG-phase-aware features (DCTR, PHARM, and features extracted using Gabor filters) were described in [81; 82; 83].

Other improvements that followed was the application of non-linear feature transformation of features [84].

The concept of calibrating features for JPEG steganalysis was introduced specifically to attack F5 [85]. It was later generalized [86], and attempts were made to port calibration to the spatial domain [13].

Feature based steganalysis generally follows the same design pattern:

- A noise extraction step which suppresses the image content, using high pass filters such as the KV filter (Equation 1.4.2), KB filter (Equation 1.4.2), Sobel filter, wiener filter, etc. [65; 87]
- A quantization step which narrows the noise residual's dynamic range,
- Co-occurence matrices formation in the horizontal, vertical, diagonal directions,
- Classifier [88; 70; 73; 89; 72], trained on the collected dataset $D(\mathcal{X}, \mathcal{Y})$

$$g_{\text{KV}} = \begin{bmatrix} -.0833 & +.1667 & -.1667 & +.1667 & -.0833 \\ +.1667 & -.5000 & +.6667 & -.5000 & +.1667 \\ -.1667 & +.6667 & -1.0000 & +.6667 & -.1667 \\ +.1667 & -.5000 & +.6667 & -.5000 & +.1667 \\ -.0833 & +.1667 & -.1667 & +.1667 & -.0833 \end{bmatrix} \tag{1.4.2}$$

$$g_{\text{KB}} = \begin{bmatrix} -1 & +2 & -1 \\ +2 & -4 & +2 \\ -1 & +2 & -1 \end{bmatrix} \tag{1.4.3}$$

More recent steganalysis detectors were built using Deep Neural Networks (DNNs) with the goal to blend in the feature extraction algorithm g into f_θ as much as possible. In other terms, learn the

classifier and the feature extraction jointly. Most work focused on Convolutional Neural Networks (CNNs) for their reduced computational cost and their suitability for image processing [90]. CNNs were first used in steganalysis in [91] with a stacked convolutional auto-encoder network. A CNN with a fixed KV filter preprocessing step was first used in [92], the authors observed that the fixed high pass filter layer was necessary to train their proposed network.

XuNet [93] also used a fixed high pass filter layer. YeNet [94] which improved the detection power drastically over XuNet, also had manually initialized filters in the first layer with SRM filters. The first Residual Neural Network (ResNet) applied to steganalysis [95] also used the fixed KV filter as a preprocessing layer and also improved upon the XuNet.

For JPEG domain steganalysis CNNs were prepended with the DCT basis as a feature extraction layer [96; 97]. Splitting features by their JPEG phase was also used in the JPEG phase-aware networks [98].

SRNet [99] was the first CNN to achieve state of the art performance in both JPEG and spatial domain steganography without any domain specific tricks besides the architectural design of preserving the input resolution using several layers of unpooled convolutions, which proved very powerful. Later work focused on making more efficient CNNs for steganalysis using more efficient design choices [100; 101; 102], or by pruning existing architectures [103].

Neural Architecture Search (NAS) [104] was also successfully used to automatically design CNN architectures [105; 106], but at the expense of a very large computational cost.

While most proposed DNN steganlyzers were tested by training on a single cover/stego source $D(\mathcal{X}, \mathcal{Y})$ defined by a cover dataset, stego scheme and payload, other work studied the effect of diversifying the stego source by creating a dataset using multiple stego schemes [107; 108], or diversifying the cover source by training detectors on multiple JPEG quality factors [109], or multiple image sizes [110; 111].

Careful thoughts were also given to data augmentation for steganalysis [112; 113; 114; 115] and the effect of the dataset size on performance [116].

The first failure of vanilla CNNs in JPEG steganalysis was discovered in [1], where CNNs were consistently underperforming for detection of nsF5 [117; 118]. The proposed OneHotConv was designed to operate directly on the DCT coefficients and overcome the failures of CNNs trained on decompressed JPEG images.

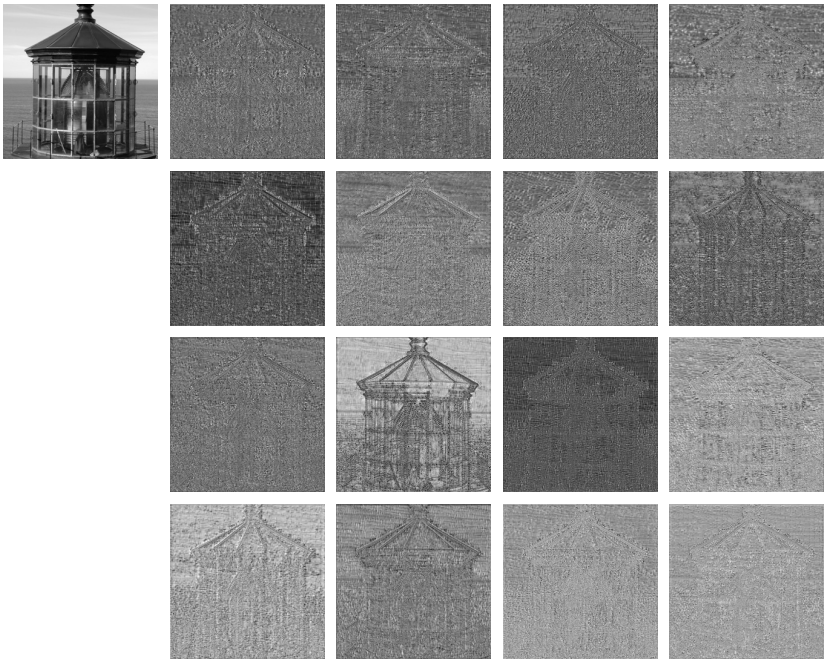


Figure 1.4.1: Outputs of SRNet's Layer 7 visualized as 16 grayscale images.

More recently, another family of DNN steganalyzers was adopted by the literature, networks pretrained on ImageNet [2; 119; 120; 121] after their success in the ALASKA II Kaggle competition [43]. These networks were originally designed for computer vision tasks and pretrained on the large ImageNet dataset [122] (1.2M images belonging to 1,000 mutually exclusive classes). ImageNet pretrained CNNs were also successful in image forensics tasks [123; 124; 125; 126].

In [2] the authors also observed that the Pair Constraint (enforcing the mini batches to include pairs of cover-stego images while training the network), previously seen as an essential element for steganalysis DNN training, was not needed when using carefully pretrained models. Moreover, not using it improved performance of their detectors.

CNNs are seen today as a superior detector family. Unlike Rich Models, the feature extraction and the classification are jointly learned end-to-end. In essence, CNNs extract noise residuals and build detection on top of them. Figure 1.4.1 shows the output of SRNet's Layer 7 visualized as 16 grayscale images. Notice how the outputs resemble noise residuals and edge detectors.

CNNs are also believed to see local artifacts introduced by steganography, unlike Rich Models (histogram based) which only leverage the effect of steganography in it's integrated form over the

entire image. This claim has been experimentally studied in [127].

1.5 Digital Images

1.5.1 The camera model

Given an incident light intensity $\mathbf{I}$, the raw pixel values $\mathbf{X}$ follow the camera model

$$\mathbf{X} = (1 + \mathbf{K})\mathbf{I} + \mathbf{\Lambda} + \mathbf{\Theta} \tag{1.5.1}$$

where $\mathbf{K}$ is the PRNU [128; 129; 130], Λ is a combination of other noise sources including the dark current, shot noise, and read-out noise, and $\mathbf{\Theta}$ is the quantization noise [131].

The raw image $\mathbf{X}$ is then fed the processing pipeline to produce the final image. The processing pipeline in digital cameras varies significantly with different camera types and manufacturers, which makes digital images a great source of cover objects, due to their complexity. A typical image processing pipeline includes signal quantization, white balance, demosaicking, color correction, gamma correction, filtering, and, usually, JPEG compression.

The strongest noise source in steganography and steganalysis applications is the shot noise. It is usually approximated using the heteroscedastic model [132; 133]. In order to be undetectable, steganography tries to preserve the statistical properties of the image noise. Natural Steganography [134; 135] uses the heteroscedastic model explicitly to hide a message by simulating a switch between two ISO settings (but requires the raw image, and strong assumptions on the processing pipeline).

The effect of various processing operations on steganographic security was studied in [44].

1.5.2 JPEG compression

JPEG stands for the Joint Photographic Experts Group that published a format standard in 1992 [136]. It is by far the most used image file format due to its widespread support.

The JPEG compression operations are done on non-overlapping 8×8 blocks of the image. For simplicity, the equations will use the notation $\mathbf{x} = (x_{ij})$ for one specific 8×8 block of a grayscale image. During JPEG compression, the DCT coefficients, $d_{kl} \in \mathbb{R}$, are obtained using the 2 dimensional

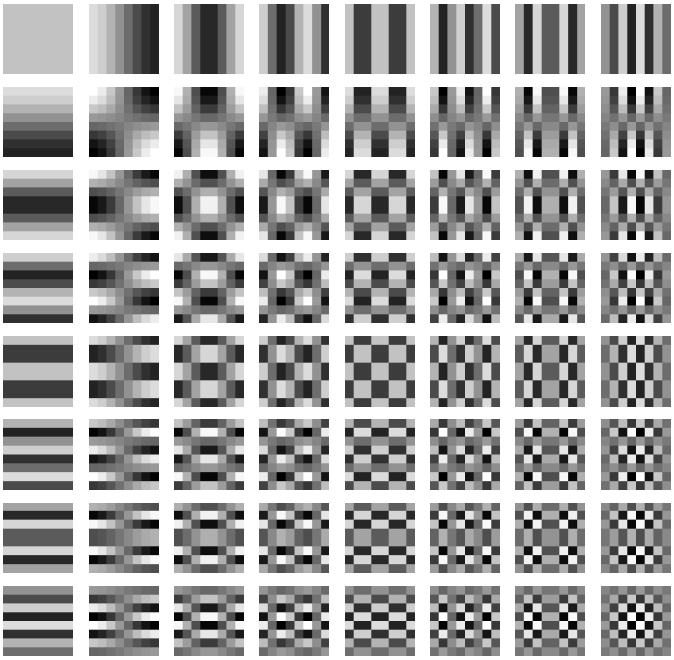


Figure 1.5.1: The DCT basis.

DCT transform

$$d_{kl} = \text{DCT2}(\mathbf{x}) \quad \triangleq \quad \sum_{i,j=0}^{7} f_{kl}^{ij}(x_{ij} - 128)\, 0 \leq k, l \leq 7, \tag{1.5.2}$$

$$f_{kl}^{ij} \quad = \quad \frac{w_k w_l}{4} \cos\frac{\pi k(2i+1)}{16} \cos\frac{\pi l(2j+1)}{16}, \tag{1.5.3}$$

where $w_k = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } k = 0 \\ 1, & \text{otherwise} \end{cases}$,

The DCT coefficients are then quantized

$$c_{kl} = [d_{kl}/q_{kl}] \tag{1.5.4}$$

where q_{kl} are the quantization steps. The quantization steps are stored in in a 8×8 quantization matrix, which is supplied in the header of the JPEG file. Because the JPEG standard allows arbitrary quantization tables to be used, as long as they are stored in the header of the JPEG file, engineers and camera makers are free to create their own. However, the JPEG standard recommends a set of

quantization matrices parametrized by a quality factor $Q \in \{1, 2, ..., 100\}$:

$$\mathbf{q}(Q) = \begin{cases} \max\{\mathbf{1}, \text{round}(2(1 - Q/100) \cdot \mathbf{q}(50))\} & Q > 50 \\ \min\{255 \cdot \mathbf{1}, \text{round}((50/Q) \cdot \mathbf{q}(50))\} & Q \leq 50 \end{cases}, \tag{1.5.5}$$

where $\mathbf{q}(50)$ is the 50% quality standard quantization table:

$$\mathbf{q}(50) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \tag{1.5.6}$$

For color images, the RGB representation is changed to YC_bC_r (luminance, and two chrominance signals) with:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{1.5.7}$$

The luminance Y is processed as above, while the chrominance signals are optionally subsampled (e.g. 4:2:2 chroma subsampling), then transformed using DCT2, and finally quantized with chrominance quantization matrices, also stored in the header of the JPEG file. The chrominance quantization steps are usually larger than luminance quantization steps because the human visual system is more sensitive to variations in brightness than color.

The JPEG compression is one of the most important step in an image processing pipeline. In fact, the JPEG compression Quality Factor controls the amount of details left in an image, Figure 1.5.3 shows that images compressed with lower quality factors have generally more zeros due to the stronger quantization. The effect of JPEG Quality Factor on steganographic security has been studied in [137] where it has been shown that the effect of the quality factor on security varies drastically from steganographic algorithm to another, the authors also predicted the trend of that effect using statistical cover models and KL divergence. Other aspects of the JPEG compression,

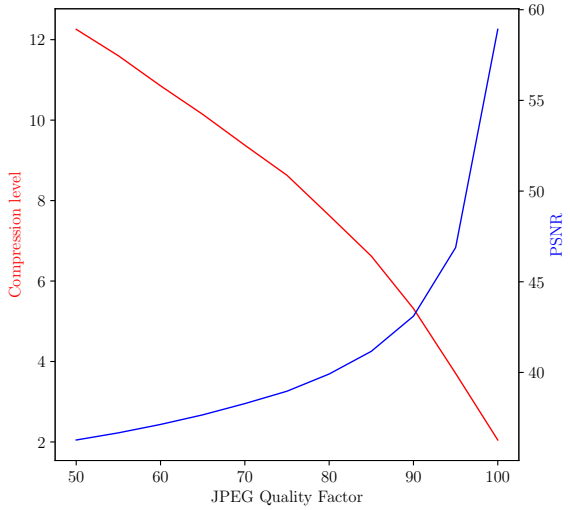


Figure 1.5.2: Peak signal-to-noise ratio (PSNR) and the compression level (original file size divided by compressed file size) between an uncompressed and compressed image using various JPEG Quality Factors.

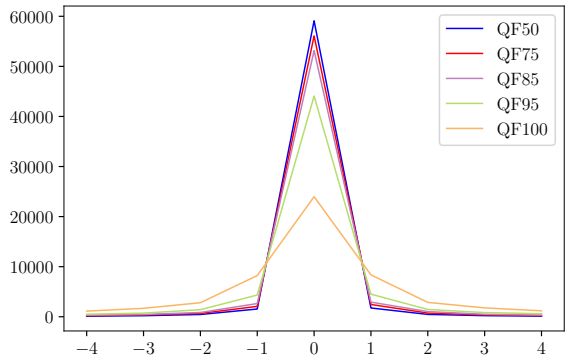


Figure 1.5.3: Histogram of DCT coefficients of a compressed image using various JPEG Quality Factors.

such as the type of integer conversion used [138] also have a great impact on steganography. Similar importance is given to the JPEG compression step in image forensics [139; 140; 141; 142].

1.6 Datasets

This section describes the datasets used in this dissertation. Unless specified otherwise, the datasets are cover datasets which can be embedded with any steganographic algorithm and payload.

1.6.1 BOSSbase+BOWS2

BOSSbase 1.01 [143] and BOWS2 [144] (BOSSbase+BOWS2) is the most popular dataset in steganography and steganalysis. Originally of size 512×512, it is usually converted to grayscale and resized to 256×256 using Matlab's 'imresize' with default parameters. The dataset was randomly divided into three sets with 14,000 (BOSSbase+BOWS2) / 1,000 (BOSSbase) / 5,000 images (BOSSbase) for training, validation, and testing.

1.6.2 ALASKA I

A total of 50,000 full size raw images made available by the ALASKA I [42] organizers. The developing script supplied by the organizers which enabled to generate several variations of the ALASKA I dataset.

The ALASKA I TILEbase was created by enforcing the smart crop to always crop "tiles" of size 256×256 instead of randomly sampling cropping sizes.

The ALASKA I ARBITRARYbase was created by using uniformly randomly sampling cropping sizes from $\{512, 640, 720, 1024\}$, similar to the final test set used by the organizers.

The training set (TRN), validation set (VAL), and test set (TST) contained respectively 42,500, 3,500, and 3,500 cover images (around 500 cover images were not used because they were corrupted or failed the processing pipeline).

1.6.3 ALASKA II

The ALASKA II [43] dataset contains $75,000$ raw images developed and processed using a script similar to the ALASKA I script. The final crop can also be modified to produce images of different sizes. We will use 2 sizes: 256×256 and 512×512.

During the ALASKA II competition, the $75,000$ images were split into $3\times25{,}000$ different cover images compressed with quality factors 75, 90, and 95.

Unlike the ALASKA I dataset, the development scripts and the stego embedding scripts were only shared after the competition ended.

The training set (TRN), validation set (VAL), and test set (TST) contains respectively $3\times22{,}000$, $3\times1{,}000$, and $3\times2{,}000$ covers.

Chapter 2

An Intriguing Struggle of CNNs in JPEG Steganalysis and the OneHot Solution

While CNN detectors [97; 145; 99] clearly outperform classifiers with hand-crafted feature sets for steganalysis in both JPEG and spatial domain (see, e. g., the detailed survey [146]), there is recent evidence that CNNs unexpectedly struggle to perform well in certain cases:

- All ALASKA steganalysis challenge participants [147; 42] consistently underperformed on nsF5 [55].
- In [137], SRNet [99] does not follow the theoretically predicted trend for nsF5 [55], while all other tested steganographic schemes follow the model.
- J-UNIWARD [19] is surprisingly best detected in JPEGs obtained with the "Trunc" quantizer [138] by JPEG rich model (JRM) [66] and not a CNN.

Figure 2.0.1 shows the total detection error under equal priors P_{E} for two scenarios in which two leading CNN architectures for JPEG domain steganalysis, the SRNet and J-XuNet [97], are outperformed by an older detection paradigm, the JRM model and the ensemble classifier [70]. In this chapter, we analyze these intriguing failures and address the deficiency with a shallow CNN, the "OneHot" CNN, that can be plugged into a conventional CNN architecture as a dual branch

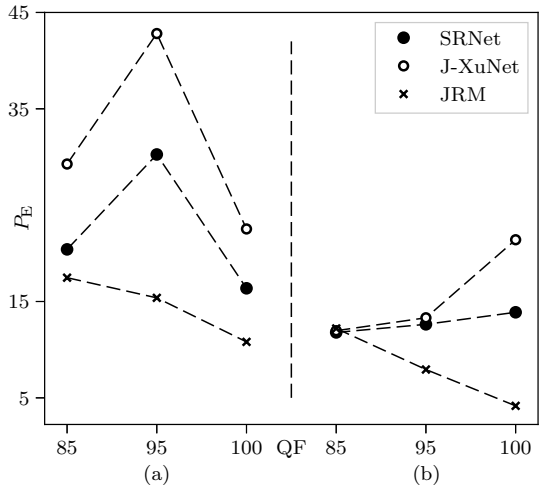


Figure 2.0.1: Detection error P_E of SRNet, J-XuNet, and JRM+ensemble for (a) J-UNIWARD 0.4 bpnzac in JPEG trunc source and (b) nsF5 0.2 bpnzac in JPEG round source.

for an end-to-end learnable detector. Methodology for plugging the "OneHot" network into conventional steganalysis CNNs is also introduced for an end-to-end learnable detector with improved performance.

In Section 2.1, we study SRNet and its variants on nsF5, and link its struggles to the inability to "see" simple artifacts in the distribution of DCT coefficients exploited by the JRM. After briefly reviewing prior art on CNNs with DCT inputs, in Section 2.2 we introduce a shallow "OneHot" CNN that can be plugged to SRNet and trained in an end-to-end fashion to address the above struggles (Section 2.3). The chapter is concluded in Section 7.7. Datasets (JPEG round/trunc sources), performance measures, and some technical aspects of training are detailed in Section 2.5.

2.1 Struggles of CNNs in JPEG steganalysis

Figure 2.0.1 shows that there exist cases where CNNs underperform by a large margin when compared to JRM, which is a rather simple feature set. Examining each JRM sub-model (Figure 2.1.1) separately reveals that most of the detection performance is due to the sub-model 'Ax_T5' corresponding to integral co-occurences from absolute values of the DCT plane computed with a clipping threshold $T = 5$. CNNs fed with decompressed JPEGs are apparently unable to see artifacts in the distribution of DCTs, such as the co-occurrence 'Ax_T5'.

Feeding the array of DCTs directly to SRNet, however, failed to produce reliable detection or did not converge (DNC). We hypothesize that this is due to the fact that, unlike pixels, DCTs are

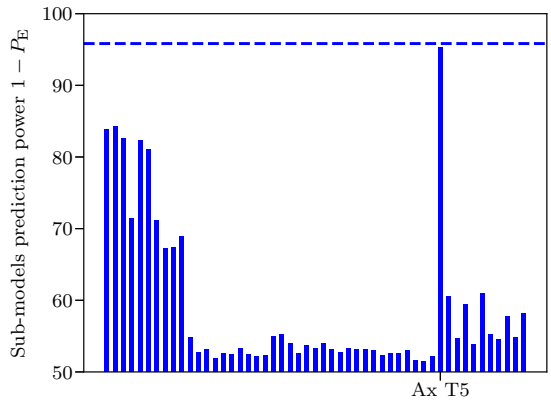


Figure 2.1.1: $1 - P_E$ when staganalyzing nsF5 0.2 bpnzac in JPEG round QF100 using individual JRM submodels and using the entire JRM (dashed line).

Table 2.1: Detection error of SRNet and its two shallower variants using the original and longer training schedule for nsF5 0.2 bpnzac JPEG round QF 100.

Architecture description	Original schedule	Longer schedule
SRNet	DNC	5.35
SRNet, Layers1-8+Global Average Pooling+FC (fully connected)	13.36	9.66
SRNet, Layers8-12+FC	25.46	19.84
JRM	**4.17**	

largely decorrelated and locally heterogeneous, making it harder for the convolutions to extract relevant image components and noise statistics. Most effort in computer vision directed towards training on DCT inputs has focused on either avoiding the costly JPEG decompression step to speed-up training [148] or on approximating a CNN trained on spatial-domain inputs [149]. Neither is relevant for our needs. In [150], a histogram layer is introduced that can compute predefined higher-order statistics, which would merely mimic the JRM.

In our case, we found out that adjusting the training schedule partially solved the problem with convergence and loss of performance at the expense of a *significantly* longer training time. Table 2.1 shows the results of SRNet with DCT inputs for nsF5 0.2 bpnzac in JPEG round QF 100 using the original training schedule [99] and a longer schedule using a doubled batch-size of 64 and seeding from a much larger payload 0.4 bpnzac. We also studied shallower versions of SRNet by pruning different layers to show that this difficulty is not linked to an excessive number of parameters to learn. Note that when using the Titan Xp GPU, SRNet's longer schedule takes 5 days to train compared to 1.2 days for the original training schedule.

2.2 OneHot CNN

A simple transformation of the input DCT plane (the "clipped one-hot encoding") before the first convolution will allow a CNN compute occurrences and co-occurrence histograms. The DCT array $\mathbf{M}$ is first clipped to a threshold T and then transformed to a binary volume of size $(T+1) \times H \times W$:

$$\begin{aligned} \mathbb{Z}^{H \times W} &\to \{0,1\}^{(T+1) \times H \times W} \\ \mathbf{M} &\mapsto \left\{ \begin{array}{ll} [|\mathbf{M}| = t], & t \in \{0, ..., T-1\} \\ [|\mathbf{M}| \geq t], & t = T \end{array} \right\} \end{aligned} \tag{2.2.1}$$

where [.] is the (element-wise) Iverson bracket [.]. In fact, one can even find a specific kernel that will compute a desired histogram. For example, horizontal co-occurrences for coefficient pairs $(x, y) \in \{0, ..., T\}^2$ can be computed by convolving the input volume with the following convolutional kernel $\mathbf{K} \in \mathbb{R}^{(T+1) \times 3 \times 3}$, followed by global average pooling

$$\mathbf{K} = \begin{cases} \mathbf{K}_t = \begin{smallmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{smallmatrix}, & t = x \\ \mathbf{K}_t = \begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{smallmatrix}, & t = y \\ \mathbf{K}_t = 0_{\mathbb{R}^{3 \times 3}} & \text{else.} \end{cases} \tag{2.2.2}$$

Inter-block statistics can be designed similarly by using dilated convolutions with rate 8 introduced in wavelet decompositions algorithms [151], also referred to as "à trous" convolutions widely used in computer vision [152; 153; 154] as a way to increase the receptive field of convolutional layers.

Figure 2.2.2 shows the overall architecture of the proposed OneHot network. Note that the "clipping" operation is necessary for memory constraints. Figure 2.2.1 shows how the detection error P_{E} reacts to different clipping thresholds. While $T = 10$ seems to be optimal, in practice any $T \geq 5$ can be chosen, as improvements recorded for higher thresholds are less than 0.5%. Tuning D_1 and D_2 also seems to have a rather minor effect on performance. However, setting $(D_1, D_1') = (64, 0)$ or $(0, 64)$, i. e., using only dilated convolutions or only vanilla convolutions, respectively, seems to hurt the detection performance by more than 2%. Table 2.2 shows that the proposed OneHot CNN performs better than JRM in the two problematic cases studied in this chapter.

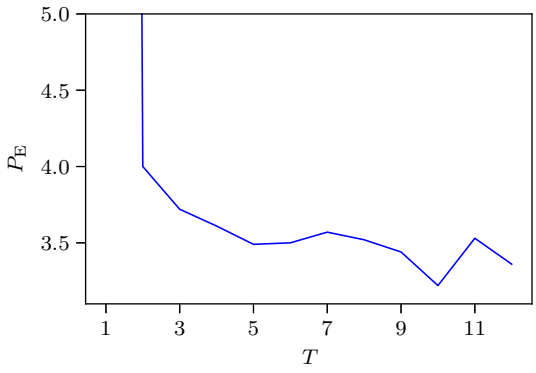


Figure 2.2.1: P_E of OneHot CNN on nsF5 0.2 bpnzac in round QF 100 JPEGs for different clipping thresholds T.

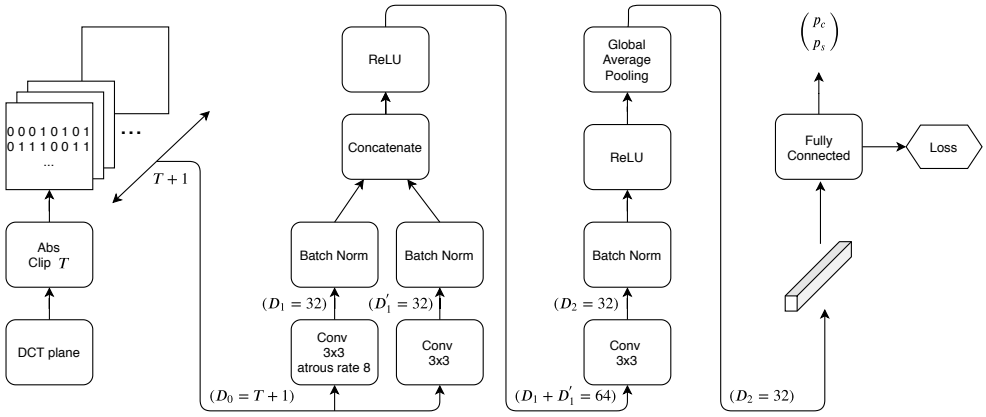


Figure 2.2.2: OneHot CNN architecture. D_i corresponds to the depth of representations at different layers of the network.

The OneHot CNN is trained with the same training schedule and hyper-parameters as SRNet, and takes around 13 hours on NVIDIA's Titan Xp GPU.

2.2.1 Cartesian Calibration

Cartesian calibration [85; 86] is a way to augment any feature set by adopting additional features computed from a reference image. The reference image is obtained by decompressing the original image, cropping by four pixels in both directions, and recompressing the cropped image. It has been shown that cartesian calibration helps nsF5, Jsteg, YASS, and other steganographic schemes [86].

We show that the OneHot CNN can also be augmented with cartesian calibration by adding a second OneHot branch taking the reference image as input. Both branches are independent until the fully FC layer, which takes a concatenation of the 2×32 representation as inputs. Denoting this

Table 2.2: Detection error P_{E} of JRM and the OneHot CNN for the two problematic cases: JPEG round, nsF5 and JPEG trunc, J-UNIWARD for various quality factors and payloads.

QF	100		95		85	
Round, nsF5	0.2	0.1	0.2	0.1	0.2	0.1
JRM	4.17	21.99	7.94	27.37	12.2	30.49
OneHot CNN	**3.49**	**20.65**	**7.90**	**27.06**	**11.28**	**29.95**
Trunc, J-UNI	0.4	0.3	0.4	0.3	0.4	0.3
JRM	10.81	19.60	15.38	23.18	17.47	24.49
OneHot CNN	**7.36**	**14.05**	**14.32**	**21.55**	**16.45**	**22.79**

Table 2.3: Detection error P_{E} of ccJRM and the calibrated OneHot CNN for JPEG round, nsF5 for various quality factors and payloads.

QF	100		95		85	
Round, nsF5	0.2	0.1	0.2	0.1	0.2	0.1
ccJRM	2.11	18.03	7.15	26.81	10.80	29.78
ccOneHot CNN	**1.39**	**15.52**	**6.66**	**25.75**	**10.09**	**29.26**

architecture ccOneHot CNN, Table 2.3 shows its superior performance w.r.t. ccJRM.

2.3 OneHot+SRNet

In this section, we show that merging the OneHot network with conventional CNN architectures produces more universal detectors. We use SRNet to show how these two networks are merged and how the resulting architecture denoted OneHot+SRNet compares to simply concatenating (a trained) SRNet's last layer and JRM features as a feature set and training FLD ensemble. This strategy is denoted JRM+SRNet.

The OneHot+SRNet is built by merging SRNet and OneHot in a branch-parallel fashion, each branch B outputs a feature extraction $\mathcal{F}_B$ (the output of the layer before FC_B, the fully connected layer of branch B) and a binary output $\mathbf{p}_B$ (the output of the FC_B+softmax). $\mathcal{F}_{SRNet}$ and $\mathcal{F}_{OneHot}$ are then concatenated and fed to the final FC layer, which outputs $\mathbf{p}_{FC}$, the final classification probability.

For each component B (SRNet, OneHot, and FC), we use the binary cross-entropy loss: $\mathcal{L}_B = -(y \log(p_B^s) + (1-y)\log(p_B^c))$, where y is the binary target, and combine the losses as follows:

$$\mathcal{L} = \sum_{B \in \{SRNet,\, OneHot,\, FC\}} \lambda_B \mathcal{L}_B, \tag{2.3.1}$$

where λ_B is a scaling hyperparameter for each branch B weighting the importance of training the

Table 2.4: Detection error P_{E} for various JPEG quality factors, round/trunc, embedding schemes, and payloads.

QF	100		95		85	
Round, nsF5	0.2	0.1	0.2	0.1	0.2	0.1
SRNet	13.88	30.76	12.63	25.35	11.70	24.39
JRM+SRNet	**1.97**	**18.11**	3.50	**19.39**	3.63	19.42
OneHot+SRNet	1.99	18.55	**3.32**	19.73	**3.50**	**19.22**
Trunc, J-UNI	0.4	0.3	0.4	0.3	0.4	0.3
SRNet	16.37	21.26	30.26	35.82	20.41	26.26
JRM+SRNet	7.62	17.49	14.32	22.87	10.14	17.76
OneHot+SRNet	**7.29**	**13.64**	**14.18**	**21.79**	**10.13**	**16.16**
Round, J-UNI	0.4	0.3	0.4	0.3	0.4	0.3
SRNet	12.52	**16.70**	**17.40**	**24.39**	9.17	14.32
JRM+SRNet	15.32	18.64	17.94	26.74	9.18	14.44
OneHot+SRNet	**11.94**	16.99	17.52	24.81	**8.84**	**14.04**
Round, UED-JC	0.3	0.2	0.3	0.2	0.3	0.2
SRNet	**6.96**	**10.16**	**10.90**	**17.56**	4.44	**7.26**
JRM+SRNet	7.69	10.53	10.99	17.86	**4.04**	7.38
OneHot+SRNet	7.26	10.58	11.35	18.25	4.40	7.58

branch B compared to other branches. The weights can be assigned manually as done in [155; 156; 157], or heuristically modeled as noise parameters and learned as done in [158]. In the following experiments we set all $\lambda_B = 1$.

Another key element in this merging architecture is making sure that each weight in the network is only updated once during back-propagation, which is done by "stopping" the gradients at the input of the merged FC layer, to ensure that the gradients of $\mathcal{L}_{FC}$ are not computed w.r.t. to the weights of SRNet and OneHot.

Table 2.3 shows that this strategy works well in practice. The first two blocks of the table show that OneHot+SRNet substantially improves SRNet. For nsF5 in JPEG round, the improvement is comparable to JRM+SRNet. For J-UNIWARD in JPEG trunc, the improvement is consistently better than JRM+SRNet. The next two blocks show that OneHot+SRNet has comparable detection performance to SRNet for J-UNIWARD and UED-JC [21] in JPEG round while avoiding some large degradations of JRM+SRNet (e. g., J-UNIWARD for QF 100 and 95).

In Table 2.5, we show that OneHot+SRNet substantially improves detection in a more diverse dataset. We use ALASKA v1 256×256 tiles as described in 2.5 compressed with JPEG quality factor 95. As prescribed by the challenge winners [147], the detectors are trained as multi-class and used as binary detectors. Color channels for SRNet are merged in the first layer by using $3 \times 3 \times 3$ convolution kernels, and the "clipped one-hot encoding" in the OneHot branch is done separately

Table 2.5: Detection error P_{E} and missed detection rate at 5% false alarm, $MD5$, for ALASKA v1 when tested against individual stego schemes and their mixture.

ALASKA v1 QF95	SRNet	OneHot+SRNet
J-UNI	10.63, 18.34	**10.97, 18.20**
EBS	**8.21, 11.51**	8.24, 11.71
UED	**10.97, 17.97**	12.04, 20.68
nsF5	27.90, 70.86	**16.37, 34.02**
Mixture	12.96, 25.08	**12.01, 20.34**

for each channel using the same threshold, producing a volume of size $3(T+1) \times H \times W$. Note that for ALASKA v1 we use $\lambda_{SRNet} = 4$ and $\lambda_{FC} = \lambda_{OneHot} = 1$ as it gave the best results.

2.4 Discussion and conclusions

While in other fields CNNs have been reported to be underperforming, for example, in solving the seemingly trivial coordinate transform problem [159], to the best of our knowledge, no prior art uncovered failings of CNNs in steganalysis. In this work, a new CNN architecture is proposed, the OneHot CNN, to overcome struggles of CNNs in at least two particular scenarios reported in this chapter. It is based on the clipped one-hot encoding, which enables computing higher-order statistics of DCT coefficients in a flexible learnable manner.

The chapter additionally describes a dual-branch architecture for adding the OneHot CNN to existing CNNs for steganalysis (SRNet) for an end-to-end trainable detector. This overcomes the reported struggles while not decreasing the performance in cases when the OneHot branch is not effective. For ALASKA v1 and QF 95, OneHot+SRNet tile detector performs 4.7% better than SRNet in terms of $MD5$ by substantially improving the detection of nsF5.

We anticipate that the proposed OneHot architecture will find applications in forensics for detection of higher-order artifacts in the distribution of DCT coefficients.

2.5 Setup of experiments

Unless mentioned otherwise, all experiments were executed on the union of BOSSbase 1.01 [143] and BOWS2 [144] converted to grayscale and resized to 256×256, the training, validation and testing splits are compatible with [99; 137]. The "trunc" and "round" sources correspond to images where the final DCT quantizer in JPEG compression is truncation towards zero and round, respectively.

In Section 2.3, ALASKA v1 [42] dataset has been used with the scripts adapted to produce 256×256 crops with JPEG compression only done in the "round" mode. This dataset was randomly divided into three sets with 42,500 / 3,500 / 3,500 for training, validation, and testing. The splits were made to be compatible with the datasets used in [147].

The steganographic algorithms used in this chapter are: J-UNIWARD [19], UED-JC [21], EBS [160], and nsF5 [55], embedded with fixed payloads (BOSS+BOWS2) or adaptive payload based on the image processing history, with priors 0.4, 0.3, 0.15, and 0.15, respectively (ALASKA v1). In ALASKA v1, color steganography is done by spreading the payload between Y, C_r, and C_b as described in [42] (*Payload repartition among color channels*).

2.6 Data augmentation in DCT domain

The first step to using DCT domain inputs in deep learning is to perform data augmentation in the DCT domain. Rotations by multiples of $\pi/2$ and horizontal/vertical flips can be done in a lossless fashion directly on the quantized DCT coefficients $\mathbf{C} \in \mathbb{Z}^{H \times W}$ thanks to the symmetries of DCT bases. We denote $f_8(\mathbf{C})$ any operation f performed in a block-wise fashion, with a block size of 8, e. g., T_8 is the 8×8 block-wise matrix transpose. For simplicity, we introduce $J = [(-1)^j]_{\substack{0 \le i < H \\ 0 \le j < W}}$ and $I = [(-1)^i]_{\substack{0 \le i < H \\ 0 \le j < W}} \in \{1, -1\}^{H \times W}$, all-ones matrices with a negative sign in odd columns and rows, respectively. Eqs. 2.6.1 and 2.6.2 show how to vertically flip and rotate by $\pi/2$

$$\text{Lossless } \mathrm{flip}^V(\mathbf{C}) = J \odot \mathrm{flip}_8^V \circ \mathrm{flip}^V(\mathbf{C}) \tag{2.6.1}$$

$$\text{Lossless } \mathrm{rot}^{\pi/2}(\mathbf{C}) = I \odot \mathrm{T}_8 \circ \mathrm{rot}_8^{3\pi/2} \circ \mathrm{rot}^{\pi/2}(\mathbf{C}), \tag{2.6.2}$$

where $\odot$ is an element-wise multiplication, and $\circ$ is the composition operation. All other valid flips and rotations can be derived in a similar fashion (or as compositions of 2.6.1 and 2.6.2).

Chapter 3

JPEG steganalysis detectors scalable with respect to compression quality

The ALASKA stegnalysis challenge [42] revealed how time and resource demanding it is to train deep learning steganalysis detectors for the "real world." A large part of this complexity is due to training detectors for each JPEG quality factor as done by the winners of the challenge [147]. This approach is not only fastidious but also not scalable to cover a potentially large number of custom quantization tables. Since deep learning architectures have shown markedly better performance than classifiers trained on hand crafted feature sets, in this chapter we explore the topic of building steganalysis detectors that would cover a wider range of quantization tables to alleviate the computational and complexity burden associated with having to train a separate detector for each quantization table.

This chapter starts by laying out preliminary definitions and notation, discussing relevant prior art and describing datasets used, steganographic schemes employed, and steganalysis tools evaluated. In Section "Scalability w.r.t. JPEG quality," we provide experimental evidence that CNN detectors as well as older feature-based detectors are scalable w.r.t. JPEG quality; quantitative comparison with dedicated detectors is given. In section "Robustness w.r.t. custom quantization tables," we look at the problem of mismatched JPEG quantization tables and show that CNN detectors trained on a range of quality factors do generalize to slightly different custom tables within the same range when measured with a semi-metric that we introduce for this purpose. In contrast, feature-based

classifiers trained on the same range of qualities appear to experience a much larger loss. Generalizing to markedly different tables remains a challenge for both types of detectors. The chapter is concluded in the last section.

3.1 Relevant prior art

The problem of mismatched JPEG quantization tables has been addressed in [161], where the authors used the 548-dimensional CC-PEV feature vector and the 22,510-dimensional CC-JRM rich model to steganalyze nsF5 [117] in different JPEG sources. The authors proposed a semi-metric comparing quantization tables, and showed that both training on a mixture of JPEG qualities as well as using the semi-metric to find the best detector from a bank of pre-trained detectors can be used in practice to steganalyze custom quantization tables without training dedicated detectors. This work does not show how those detectors trained on a mixture of qualities compare to dedicated detectors.

In [162], the authors use a kernel based feature transformation to adapt CC-PEV and CC-JRM to mismatched JPEG quantization tables. However, it is unclear how to adapt this transformation to deep learning detectors where the feature representation is learned.

The mismatch of JPEG quantization tables between training and testing sets is a special case of what is recognized as the cover source mismatch problem [163; 164]. In [49], it is shown that, for a fixed feature set (CC-300), simple classifiers, such as the FLD-ensemble, are more robust to the cover source mismatch. This begs a question of how the mismatch affects modern detectors built using deep learning, which jointly optimize the feature representation and the classifier and are thus highly non-linear.

In [108], the authors showed that the SRNet (trained as a multi-class detector) is able to contain the complexity of a diversified stego source. The findings of this paper were used by the winners of the ALASKA challenge [147] to build detectors for a more diverse stego source. Even though the ability to generalize to unseen steganographic methods is still a challenge, these results indicate that properly trained CNNs do have the capacity to deal with diverse sources.

3.2 Datasets

Most experiments in this chapter were executed on images prepared from BOSSbase 1.01 [143] and BOWS2 [144] each with 10,000 grayscale images resized to 256×256.

In Section 3.4.2 the ALASKA v1 [42] 256×256 "tiles" dataset was used.

When training detectors based on hand-crafted feature sets, the validation set is merged with the training set and a k-fold cross validation or any other prediction performance estimate can be used to determine the optimal hyper-parameters.

The steganographic algorithms used in this chapter are: J-UNIWARD [19], UED-JC [21], EBS [160], and nsF5 [117], embedded with 0.4 and 0.3 bpnzac (BOSS+BOWS2) or adaptive payload based on the image processing history, with priors $\pi_i = 0.4$, 0.3, 0.15, and 0.15, respectively (ALASKA v1). In ALASKA v1, color steganography is done by spreading the payload between between the image components (Y, C_r, and C_b) as described in Section *Payload repartition among color channels* in [42].

3.3 Steganalysis feature sets and CNN architecture

3.3.1 Feature based steganalysis

The steganalysis community has come up with numerous feature sets built in different domains and for different stego algorithms. In this chapter, we work with the popular feature set called DCTR [81].

The DCTR features are histograms of absolute values of undecimated DCT coefficients quantized by

$$q_Q = 8 \times \left(2 - \frac{Q}{50}\right). \tag{3.3.1}$$

The undecimated DCT coefficients are defined as a set of 64 convolutions indexed by (k, l) with the DCT bases f_{kl}^{ij} described in 1.5.2.

In this chapter, we extend (3.3.1) to custom quantization tables $\mathbf{q}$ as

$$q = 8 \times \bar{\mathbf{r}}, \tag{3.3.2}$$

where $\mathbf{r} = \mathbf{q} \oslash \mathbf{q}(50)$ is the elementwise division of both matrices, and $\bar{\mathbf{r}}$ corresponds to the average of all elements of matrix $\mathbf{r}$.

The FLD-ensemble [70] is then used with its default parameters for constructing the detector. When trained on multiple JPEG qualities, the training dataset corresponds to copies of the same set of images compressed with each quality.

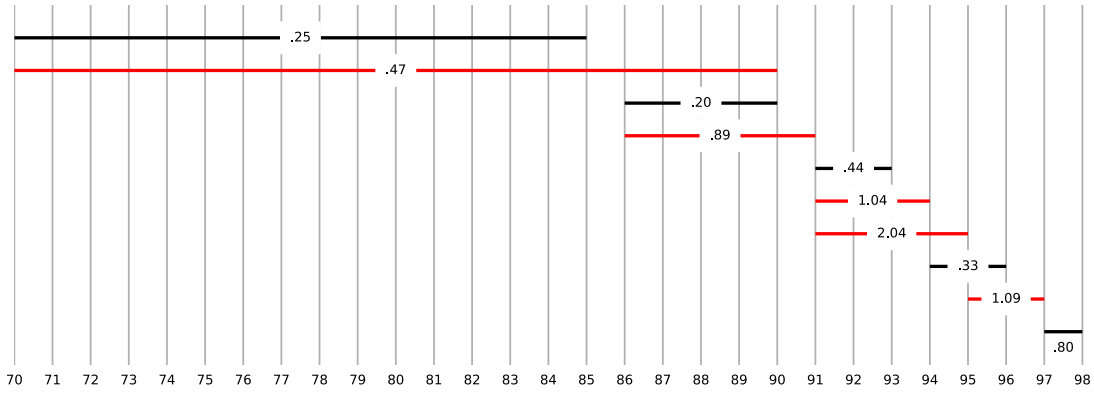


Figure 3.3.1: Maximum loss of accuracy of multi-quality SRNets w.r.t. dedicated detectors in multiple JPEG quality ranges for J-UNIWARD (0.4 bpnzac). Black lines correspond to selected ranges, widening them (red lines) leads to an increase of this loss.

3.3.2 CNN steganalysis

Recently, the community turned to deep learning for stegnanalysis in an attempt to improve detection accuracy by jointly optimizing the image representation (features sets) as well as the classifier. Deep learning architectures, such as [97; 145; 99], have been shown to outperform hand-crafted feature sets in the JPEG domain. In addition to Chapter 1, a detailed survey on deep learning in steganalysis can be found in [146].

In this chapter, we use the SRNet [99], a residual [165] CNN with 3×3 convolution kernels and ReLU activation functions. The first 8 layers of SRNet are un-pooled, and the next convolutional blocks are pooled using a 3×3 averaging layer with stride 2, as well as strided 1×1 convolutions in the skip connections. SRNet applies global average pooling in the last pooled layer to a 512 feature map, which is then Fully Connected (FC) to the classification logits. SRNet is trained with the Adamax optimizer [166] using various mini-batch sizes adapted to the diversity of the sources, as opposed to the mini-batch size of 32 initially proposed in [99]. At the time of publishing this work, SRNet achieved the best overall results for steganalysis in the JPEG domain.

When trained on multiple qualities, each batch is formed by repeatedly uniformly sampling a JPEG quality factor and selecting a cover-stego pair of that JPEG quality.

3.4 Scalability w.r.t. JPEG quality

In this section, we investigate whether feature-based and CNN steganalysis can contain the complexity of multiple JPEG quality factors within a single model. Note that quality factors 99 and 100 are not studied in this section because a very reliable JPEG compatibility attack is available for these qualities [167]. We study the scalability of this attack w.r.t. JPEG quality in 3.4.3.

Starting with small ranges, we compute the maximum loss of accuracy of the detector trained on the range w.r.t. dedicated detectors trained on individual qualities. The range is then expanded until we start observing high losses. Figure 3.3.1 shows the results of these experiments.

The selected ranges of quality factors in Figure 3.3.1 show an interesting general rule of thumb. A range of JPEG quality factors $[Q_{\min}, Q_{\max}]$ can be grouped in a single detector as long as :

$$\mathbf{q}_{kl}(Q_{\min})/\mathbf{q}_{kl}(Q_{\max}) \lesssim 2, \forall\, 0 \leq k, l \leq 7. \tag{3.4.1}$$

Note that when training SRNet on the range [70, 85] we use mini-batch size 128 due to the increased diversity introduced by mixing many JPEG qualities. All other ranges are trained using mini-batch size 64, no scaling of learning rates or the number of training iterations has been performed.

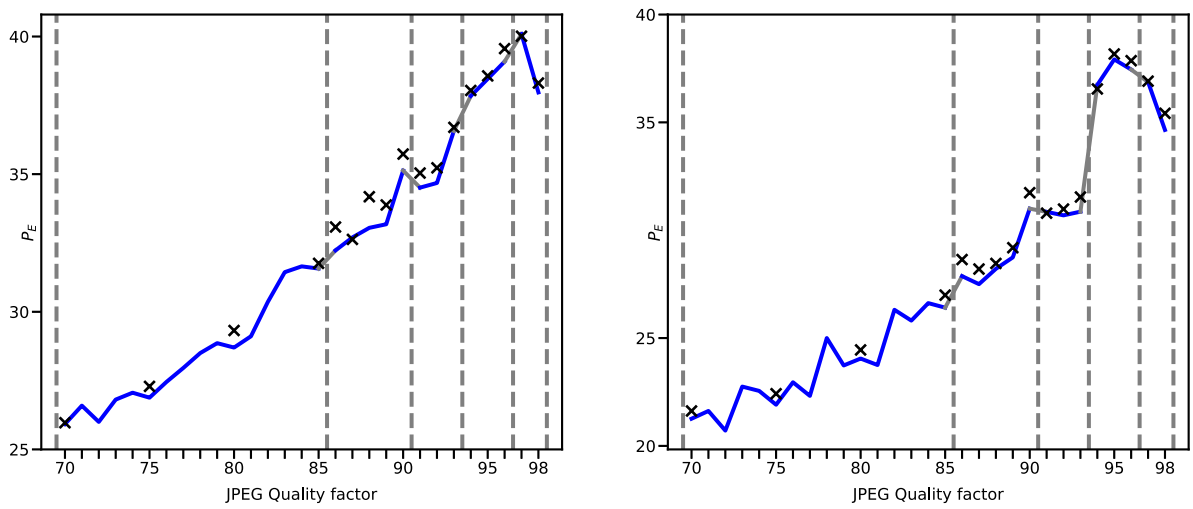
3.4.1 BOSS+BOWS2

Figure 3.4.1 shows the minimum total error probability P_{E} under equal priors for J-UNIWARD (0.4 bpnzac) and UED (0.3 bpnzac) for detectors dedicated to a specific JPEG quality (crosses) and detectors trained on each bin (range). Both DCTR+FLD-ensemble and SRNet seem to scale to multiple quality factors with no substantial loss in performance in both stego sources.

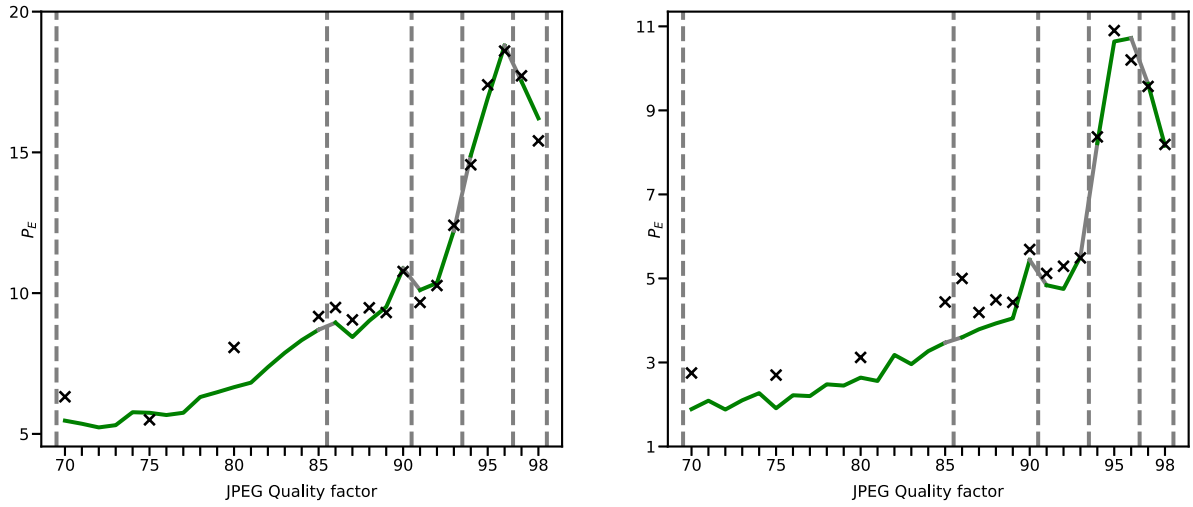
SRNet, however, has a significantly better detection accuracy that does not come at the expense of capacity of scaling to multiple qualities. The "ripples" in performance are explained in [137] and are due to the rounding and maxing in quantization matrices.

3.4.2 ALASKA

In order to move the experiments to a more realistic setting, we now use the ALASKA v1 dataset to test the proposed JPEG qualities grouping strategy. We show that the proposed grouping scales to



(a) DCTR+FLD-ensemble



(b) SRNet

Figure 3.4.1: Minimum error probability P_{E} of multi-quality detectors for J-UNIWARD (0.4 bpnzac) (left) and UED (0.3 bpnzac) (right) detectors compared to dedicated detectors using DCTR+FLD-ensemble (a) and SRNet (b). Dashed grey lines represent the bins of JPEG qualities for multi-quality detectors.

more diverse cover and stego sources as well. Figure 3.4.2 shows the minimum error probability P_{E} and MD5[1] of YC_rC_b-SRNet tile detectors (c. f., *Channel separation* in [147]), trained as multi-class and used as binary detectors as executed by the authors. The grouping strategy does not affect the detectors' performance using either performance measure. For all QF ranges, we use mini-batch size 128 due to the increased diversity of the ALASKA dataset.

Note that, unlike the figures in [147], where the authors reported on a single test set comprising the stego mixture, measures in Figure 3.4.2 are computed using the following characterization of the ROC curve. If we denote the soft-output of a detector as $\hat{y}$ and the true label as y, then:

$$\begin{aligned} P_{\mathrm{MD}}(T) &= P(\hat{y} \leq T \mid y > 0) \\ &= \sum_{i=1}^{4} P(\hat{y} \leq T \mid y = i) P(y = i \mid y > 0) \\ P_{\mathrm{MD}}(T) &= \sum_{i=1}^{4} \pi_i P(\hat{y} \leq T \mid y = i). \end{aligned} \tag{3.4.2}$$

$$\begin{aligned} P_{\mathrm{FA}}(T) &= P(\hat{y} \geq T \mid y = 0) \\ P_{\mathrm{FA}}(T) &= 1 - P(\hat{y} \leq T \mid y = 0). \end{aligned} \tag{3.4.3}$$

This gives a more robust estimation of performance measures as P_{MD} is computed over stego versions of all available TST covers instead of only a portion of TST covers for each stego scheme.

3.4.3 Reverse JPEG compatibility attack

The reverse JPEG compatibility attack is a powerful universal steganalysis attack for quality factors 99 and 100. In [167], the authors explain that for JPEG QF99 and QF100, the best detectors are built by training on the rounding errors of decompressed images instead of the images themselves. In particular, they replace the inputs of SRNet with the rounding errors to get the best detectors. Table 3.1 shows that for these two qualities, SRNet trained on rounding errors is also scalable w.r.t. the diversification.

[1] Missed detection rate at 5% false alarm.

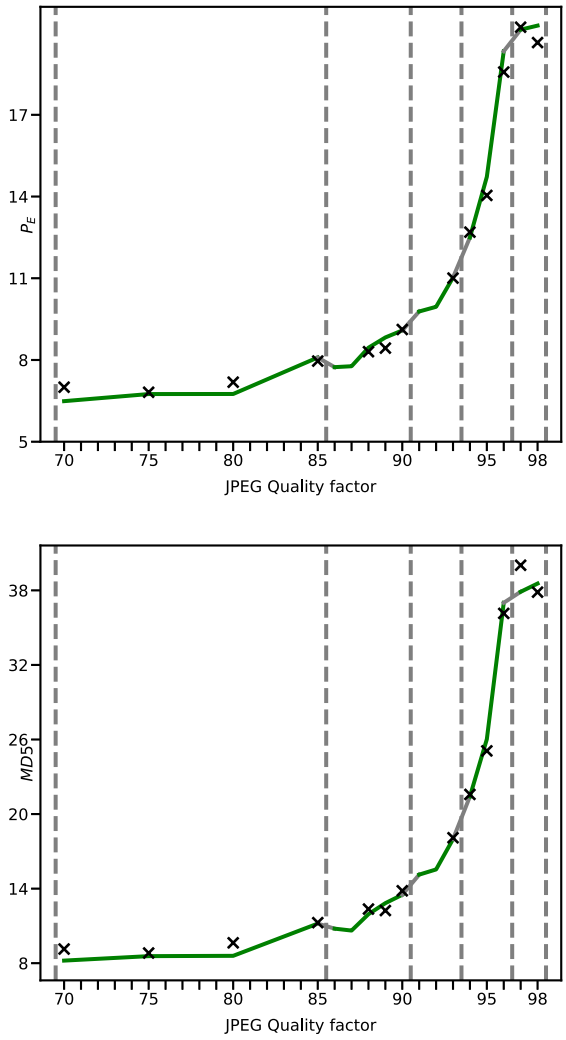


Figure 3.4.2: Minimum error probability P_{E} and missed detection rate at 5% false alarm MD5 of multi-quality detectors for ALASKA v1 compared to dedicated detectors trained during the competition. Dashed grey lines represent the bins of JPEG qualities for multi-quality detectors.

QF	100		99	
Payload	0.1	0.05	0.1	0.05
Dedicated	0.02	0.54	6.84	20.11
Trained on QF99–100	0.09	0.43	6.96	19.41

Table 3.1: Minimum error probability P_{E} of multi-quality detectors for J-UNIWARD compared to dedicated detectors trained using of the reverse JPEG compatibility attack.

3.5 Robustness w.r.t. custom quantization tables

In this section, we selected 14 custom quantization tables from various camera models. The goal is to investigate the ability of both detection paradigms to generalize to unseen custom JPEG quantization tables.

3.5.1 A semi-metric comparing quantization tables

We introduce the following semi-metric to compare two quantization tables $\mathbf{q}$ and $\mathbf{p}$:

$$d^2(\mathbf{q}, \mathbf{p}) = \sum_{k,l} \frac{1}{(k+l)^2} \left(\frac{\mathbf{q}_{kl} - \mathbf{p}_{kl}}{\mathbf{q}_{kl} + \mathbf{p}_{kl}} \right)^2, \tag{3.5.1}$$

which is a weighted sum of the squares of relative differences between corresponding quantization coefficients. The weights are larger for low spatial frequencies (upper left of the table) and lower for high spatial frequencies (lower right of the table). We refer to it as the quantization table "dissimilarity" measure. It allows us to link each quantization table to the nearest JPEG quality: $\hat{Q}(\mathbf{q}) = \text{argmin}_Q\, d(\mathbf{q},\mathbf{q}(Q))$. For a quantization table $\mathbf{q}$, we denote $\mathcal{B}(\mathbf{q})$ as the bin of JPEG qualities (used for training) to which $\hat{Q}(\mathbf{q})$ belongs. For notational simplicity, we denote the P_{E} obtained when training on $\mathbf{p}$ (one or multiple standard or custom quantization tables) and testing on $\mathbf{q}$ as $P_{\text{E}}(\mathbf{p}, \mathbf{q})$.

Figure 3.5.1 shows how the dissimilarity relates to SRNet's performance when the quantization tables are mismatched: the minimums are synchronized or sometimes relatively flat around the optimal values. This shows that the $\hat{Q}(\mathbf{q})$ computed using the proposed dissimilarity measure will usually be the best JPEG quality to steganalyze with, i. e., $\hat{Q}(\mathbf{q}) = \text{argmin}_Q\, d(\mathbf{q},\mathbf{q}(Q)) = \text{argmin}_Q\, P_E(\mathbf{q}(Q), \mathbf{q})$, which is a desirable property of the dissimilarity measure.

Table 3.3 shows the P_{E} of SRNet and DCT+FLD-ensemble in different settings. Each row corresponds to a custom quantization table $\mathbf{q}$. These results are visualized in Figure 3.5.2, which shows that SRNet is markedly more robust to mismatched custom quantization tables. Figure 3.5.2 also shows that training on multiple JPEG qualities does not seem to affect this robustness on average. Finally, Figure 3.5.2 points at some irregular behavior of the dissimilarity measure proposed. Even though, in most cases, a large dissimilarity value implies a larger loss in P_{E}, these losses are not consistently decreasing as a function of the dissimilarity.

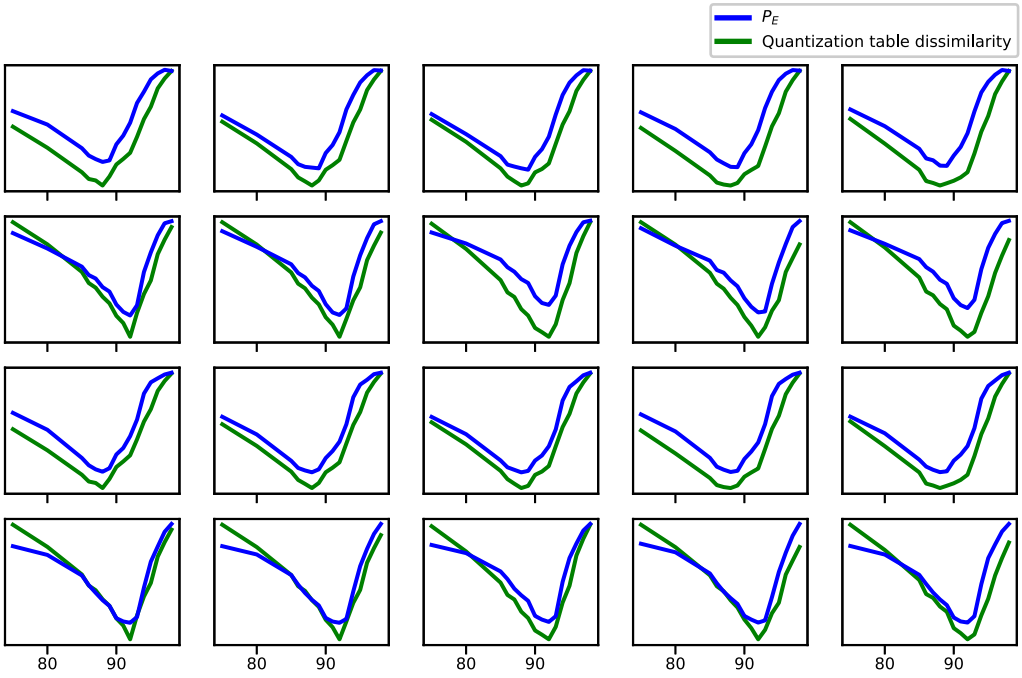
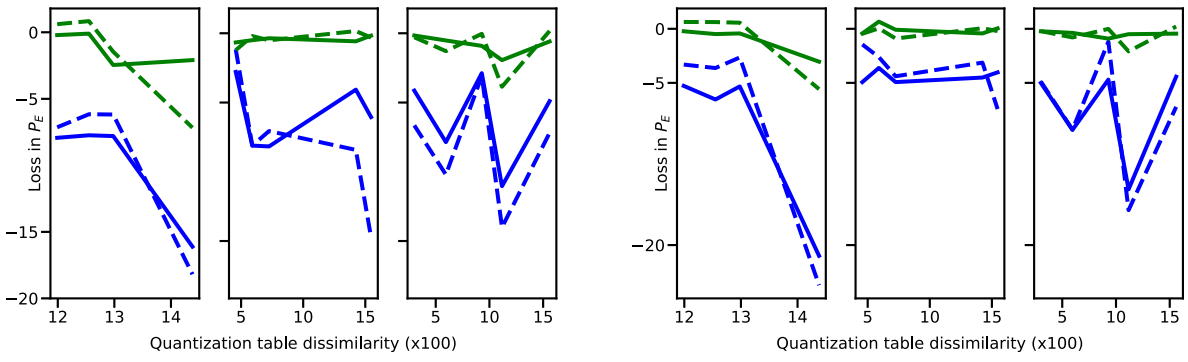


Figure 3.5.1: Minimum error probability of SRNet $P_{\mathrm{E}}(\mathbf{q}(Q), \mathbf{q})$ and $d(\mathbf{q}(Q), \mathbf{q})$ for different quality factors Q ranging between 75 and 98 and for 10 custom quantization tables $\mathbf{q}$. The first two rows correspond to J-UNIWARD (0.4 bpnzac), while the rest correspond to UED (0.3 bpnzac).



(a) J-UNIWARD (0.4 bpnzac)

(b) UED (0.3 bpnzac)

Figure 3.5.2: Loss in P_{E} for custom quantization tables: $P_{\mathrm{E}}(\mathbf{q}, \mathbf{q}) - P_{\mathrm{E}}(\mathbf{q}(\hat{Q}), \mathbf{q})$ in solid, $P_{\mathrm{E}}(\mathbf{q}, \mathbf{q}) - P_{\mathrm{E}}(\mathcal{B}(\mathbf{q}), \mathbf{q})$ in dashed for SRNet (green) and DCTR+FLD-ensemble (blue). Subplots refer to the three groups of quantization tables in Table 3.3.

3.6 Numerical results

Table 3.2 shows examples of custom quantization tables used in this chapter, their closest standard quantization table and their dissimilarity measure.

3.7 Conclusions

This chapter investigates the problem of detecting steganography in a diverse cover source of JPEG images. We are particularly interested in how steganalysis detectors scale to multiple JPEG qualities within a single model training, which we refer to as scalability w.r.t. JPEG quality factors. We show that both feature-based and CNN based detectors scale to multiple quality factors. We propose a set of detectors trained on a mixture of quality factors which, when compared with dedicated detectors trained for a specific JPEG quality factor, show no substantial loss in performance. The mixtures have been developed by gradually adding quality factors until a loss is observed when compared to dedicated detectors.

A set of 14 custom quantization tables with various dissimilarity measures to standard tables has been used to experimentally demonstrate that the scalability w.r.t. multiple JPEG qualities does not come at the expense of the detectors' robustness when facing mismatched custom quantization tables. CNN based steganalysis show a markedly better robustness compared to feature-based detectors.

This chapter's general outcome is that we do not need to train a detector for each quality factor. This is very useful in practice, where one inevitably faces a diverse JPEG cover source, as it was the case, for example, in the ALASKA challenge.

Custom Quantization Table								Standard Quantization Table								$\hat{Q}$	$d(\mathbf{q}, \mathbf{q}(\hat{Q}))$ ($\times 100$)
8	6	7	7	9	12	25	36	8	6	5	8	12	20	26	31		
6	6	7	9	11	18	32	46	6	6	7	10	13	29	30	28		
5	7	8	11	19	28	39	48	7	7	8	12	20	29	35	28		
8	10	12	15	28	32	44	49	7	9	11	15	26	44	40	31	75	11.99
12	13	20	26	34	41	52	56	9	11	19	28	34	55	52	39		
20	29	29	44	55	52	61	50	12	18	28	32	41	52	57	46		
26	30	35	40	52	57	60	52	25	32	39	44	52	61	60	51		
31	28	28	31	39	46	51	50	36	46	48	49	56	50	52	50		
8	8	8	8	8	17	25	25	8	6	5	8	12	20	26	31		
8	8	8	8	8	25	25	25	6	6	7	10	13	29	30	28		
8	8	8	8	17	25	25	25	7	7	8	12	20	29	35	28		
8	8	8	8	25	33	33	25	7	9	11	15	26	44	40	31	75	14.38
8	8	17	25	25	42	42	33	9	11	19	28	34	55	52	39		
8	17	25	25	33	42	50	42	12	18	28	32	41	52	57	46		
17	25	33	33	42	50	50	42	25	32	39	44	52	61	60	51		
33	42	42	42	50	42	42	42	36	46	48	49	56	50	52	50		
4	3	2	4	6	11	14	16	4	3	2	4	6	10	12	15		
3	3	3	5	7	16	16	15	3	3	3	5	6	14	14	13		
3	3	4	6	11	15	19	15	3	3	4	6	10	14	17	13		
3	4	6	8	14	24	22	17	3	4	5	7	12	21	19	15	88	4.57
4	6	10	15	18	30	28	21	4	5	9	13	16	26	25	18		
6	9	15	17	22	28	31	25	6	8	13	15	19	25	27	22		
13	17	21	24	28	33	33	27	12	15	19	21	25	29	29	24		
19	25	26	27	30	27	28	27	17	22	23	24	27	24	25	24		
3	3	3	3	8	5	13	13	4	3	2	4	6	10	12	15		
3	3	3	3	5	11	13	13	3	3	3	5	6	14	14	13		
3	3	5	5	5	11	16	16	3	3	4	6	10	14	17	13		
3	5	5	8	13	16	24	24	3	4	5	7	12	21	19	15	88	15.46
3	5	8	13	16	18	13	26	4	5	9	13	16	26	25	18		
11	8	13	16	18	24	18	24	6	8	13	15	19	25	27	22		
5	16	21	18	24	26	24	21	12	15	19	21	25	29	29	24		
16	18	21	21	26	26	24	24	17	22	23	24	27	24	25	24		
3	2	2	3	4	3	8	9	3	2	2	3	4	6	8	10		
2	2	2	3	4	9	9	8	2	2	2	3	4	9	10	9		
2	2	3	4	6	9	10	8	2	2	3	4	6	9	11	9		
2	3	4	5	8	13	12	9	2	3	4	5	8	14	13	10	92	3.01
3	4	6	8	10	16	15	11	3	4	6	9	11	17	16	12		
4	5	8	9	12	15	16	13	4	6	9	10	13	17	18	15		
7	9	11	13	15	18	17	15	8	10	12	14	16	19	19	16		
11	13	14	14	16	15	15	14	12	15	15	16	18	16	16	16		
2	2	2	2	6	4	9	9	3	2	2	3	4	6	8	10		
2	2	2	2	4	8	9	9	2	2	2	3	4	9	10	9		
2	2	4	4	4	8	13	11	2	2	3	4	6	9	11	9		
2	4	4	6	9	11	17	17	2	3	4	5	8	14	13	10	92	15.60
2	4	6	9	11	13	9	19	3	4	6	9	11	17	16	12		
8	6	9	11	13	17	13	17	4	6	9	10	13	17	18	15		
4	11	15	13	17	19	17	15	8	10	12	14	16	19	19	16		
11	13	15	15	19	19	17	17	12	15	15	16	18	16	16	16		

Table 3.2: Examples of custom quantization tables used and their closest standard counterparts.

$\hat{Q}$	$d(\mathbf{q},\mathbf{q}(\hat{Q}))$ (x100)	J-UNIWARD (0.4 bpnzac) $P_{\mathrm{E}}(\mathbf{q},\mathbf{q})$	$P_{\mathrm{E}}(\mathcal{B}(\mathbf{q}),\mathbf{q})$	$P_{\mathrm{E}}(\mathbf{q}(\hat{Q}),\mathbf{q})$	UED (0.3 bpnzac) $P_{\mathrm{E}}(\mathbf{q},\mathbf{q})$	$P_{\mathrm{E}}(\mathcal{B}(\mathbf{q}),\mathbf{q})$	$P_{\mathrm{E}}(\mathbf{q}(\hat{Q}),\mathbf{q})$
75	11.99	6.42	5.81	6.63	2.87	2.24	3.10
	12.55	6.83	5.99	6.93	3.14	2.52	3.63
	12.98	4.47	5.93	6.93	3.05	2.49	3.48
	14.38	6.91	14.08	9.00	1.35	6.93	4.39
88	4.57	10.58	11.80	11.24	4.66	5.12	5.09
	5.90	8.57	8.77	9.07	4.28	4.24	3.62
	7.25	8.39	8.90	8.75	3.41	4.31	3.50
	14.24	8.83	8.67	9.41	3.99	3.94	4.42
	15.46	9.66	9.97	9.83	4.09	4.30	4.05
92	3.01	9.84	10.13	10.00	5.10	5.32	5.33
	5.95	9.61	10.90	10.13	4.72	5.50	5.10
	9.29	13.10	13.14	14.01	6.63	6.64	7.54
	11.17	9.00	12.86	10.94	4.05	6.14	4.55
	15.60	12.28	12.11	12.87	5.59	5.38	6.04

(a) SRNet

$\hat{Q}$	$d(\mathbf{q},\mathbf{q}(\hat{Q}))$ (x100)	J-UNIWARD (0.4 bpnzac) $P_{\mathrm{E}}(\mathbf{q},\mathbf{q})$	$P_{\mathrm{E}}(\mathcal{B}(\mathbf{q}),\mathbf{q})$	$P_{\mathrm{E}}(\mathbf{q}(\hat{Q}),\mathbf{q})$	UED (0.3 bpnzac) $P_{\mathrm{E}}(\mathbf{q},\mathbf{q})$	$P_{\mathrm{E}}(\mathcal{B}(\mathbf{q}),\mathbf{q})$	$P_{\mathrm{E}}(\mathbf{q}(\hat{Q}),\mathbf{q})$
75	11.99	27.48	34.62	35.41	22.85	26.16	28.13
	12.55	28.2	34.36	35.94	24.51	28.14	31.05
	12.98	27.47	33.65	35.27	23.41	26.06	28.74
	14.38	23.73	41.92	39.84	15.79	39.5	36.78
88	4.57	35.82	36.98	38.61	30.43	31.88	35.36
	5.90	33.47	41.58	41.58	27.44	30.06	31.06
	7.25	33.18	40.24	41.35	27.15	31.54	32.08
	14.24	34.22	42.64	38.29	28.38	31.51	32.89
	15.46	34.02	48.74	40.07	27.24	34.66	31.27
92	3.01	35.24	41.89	39.38	30.64	35.52	35.65
	5.95	34.98	45.19	42.84	29.83	39.16	39.18
	9.29	37.88	40.96	40.76	34.35	35.55	39.05
	11.17	33.85	47.86	44.89	28.34	45.11	43.19
	15.60	37.07	44.26	41.97	31.92	39.14	36.35

(b) DCTR+FLD-ensemble

Table 3.3: Minimum total error probability P_{E} of various detectors: (i) dedicated $P_{\mathrm{E}}(\mathbf{q},\mathbf{q})$ (ii) trained on the corresponding bin $P_{\mathrm{E}}(\mathcal{B}(\mathbf{q}),\mathbf{q})$ (iii) trained on the closest JPEG quality $P_{\mathrm{E}}(\mathbf{q}(\hat{Q}),\mathbf{q})$, using SRNet (a) and DCTR+FLD-ensemble (b). Each row corresponds to a custom quantization table.

Chapter 4

ImageNet Pre-trained CNN Models for JPEG Steganalysis

In this chapter, we share our experience with recent computer-vision models originally pre-trained on ImageNet [122] for image classification, which were refined for steganalysis in the JPEG domain. This approach was predominantly employed by virtually all top performers during the recent steganalysis competition ALASKA II hosted on Kaggle. Pre-training exposes the CNN to more than a million images from a very large number of sources, extremely diverse processing, and diverse content. As such, the filters in their convolutional layers are able to recognize a great diversity of shapes, textures, noise patterns, processing and image-development traces, which are exactly the attributes that modulate the stego signal of modern content-adaptive steganographic schemes. Detecting stego noise is essentially equivalent to detecting traces of the content itself.

This approach is fairly new to the steganalysis literature. In fact, to the best of our knowledge, only one published paper [168] uses an ImageNet pre-trained model for steganalysis in spatial domain but does not compare it to any other steganalysis detector, nor uses a standard dataset.

In the next section, we introduce ImageNet models and methods of transfer learning for steganalysis. Section 4.2 describes the experimental setup, Section 4.3.1 lays out the chapter's main experimental results, Section 4.4 briefly describes the ALASKA II competition and our prize winning submission. The chapter is concluded in Section 7.7.

4.1 ImageNet models

ImageNet is one of the largest computer vision benchmark databases. Most computer vision research uses the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) "trimmed" version of 1,000 classes and approximately 1.3 million training images.

This chapter experiments with CNNs originally trained on ImgeNet and with a model scaling parameter that controls the size of the network: ResNet [165] and its variants, TResNet [169], SK-ResNeXt [170], and DenseNet [171], as well as the EfficientNet [172] and MixNet [173] (MN).

In addition to recent advances in neural architectures benchmarked on ImageNet, a large body of work is dedicated to transfer learning, i. e., using ImageNet pre-trained neural networks and fine-tuning either the entire network or a subset of the network on a new task. A recent study of transfer learning of ImageNet models [174] shows that better ImageNet models transfer better to new tasks.

4.1.1 Steganalysis transfer learning

Steganalysis transfer learning was done using two pipelines: (A) A pipeline inspired by Alex Shonenkov's public baseline,[1] using cross-entropy loss and the AdamW optimizer with 10^{-2} weight decay, for 50 epochs using "ReduceLROnPlateau" Learning Rate (LR) scheduler monitoring the validation loss with a start LR of 10^{-3}, a patience of 2 epochs, a multiplier of 1/2, and D4 training augmentation. (B) A slightly different pipeline using a multi-head classifier (Binary and Multi-class heads) and cosine annealing LR decay from 10^{-3} to 10^{-5}, for 100 epochs, and using coarse dropout with a small probability and a maximum number of zeroed regions set to 1, in addition to D4 augmentations. All training augmentations were performed using the Albumentations library [175].

Targeted experiments indicated that the most influential hyper-parameters in fine-tuning ImageNet models using both pipelines are the LR and the weight decay, which have to be adjusted for each optimizer. Due to time constraints, we did not perform a rigorous search for optimal settings but early experiments showed that (A) and (B) worked rather well for various CNNs although (B) gave slightly better results for larger architectures.

Networks larger than 10M parameters were trained using Automatic Mixed Precision (AMP), from NVIDIA's apex library.

[1] `https://www.kaggle.com/shonenkov/train-inference-gpu-baseline`

Note that the refining detailed here has been used on the ALASKA II training dataset described in Section 4.2. It is possible that it may need to be adjusted for best results when refining other pre-trained models on other datasets, stego methods, and a different set of JPEG quality factors.

4.2 Experimental setup

The majority of experiments reported upon in this chapter were executed on the training dataset made available by the organizers of ALASKA II. It contains 3×25,000 different cover images compressed with quality factors 75, 90, and 95, and the same amount of stego images embedded with J-UNIWARD [19], J-MiPOD [176], and UERD [21], making the training set size 4×75,000 images. Per the description of the organizers, the payload embedded in each image was scaled so that all images are approximately equally difficult to detect, with comparatively smaller payload embedded in smooth images and larger payloads embedded in highly textured or noisy images. The average payload embedded across the database was 0.4 bits per non-zero AC DCT coefficient (bpnzac).

Unless mentioned otherwise, for the purpose of the competition the training set was randomly split into three disjoint subsets while making sure each cover image was in the same subset as its three stego versions: the training, validation, and testing sets with 4×3×22,000, 4×3×1,000, and 4×3×2,000 images, respectively.

Most experiments were evaluated with a performance measure derived from the receiver operating characteristic curve (ROC) defined as the probability of correct detection of stego image as a function of the probability of false alarm, $P_{\mathrm{D}}(P_{\mathrm{FA}})$: the weighted area under the ROC (wAUC) used in ALASKA II.

Experiments in Section 4.3.4 are evaluated using the missed detection rate at false alarm 0.05 MD5, and the minimum average total error under equal priors P_{E}, for consistency with the ALASKA I competition results.

4.3 Results

4.3.1 Baseline

As a baseline for the current state of the art in steganalysis of JPEG images, we selected the three-channel SRNet [99] ($YCrCb$), an architecture specifically designed for steganalysis. It was

Model	UERD			J-UNIWARD			J-MiPOD			Mixture
QF	75	90	95	75	90	95	75	90	95	
SRNet	0.9208	0.9081	0.8987	0.8675	0.8459	0.8499	0.9760	0.9604	0.8199	0.8934
SRNet noPC	0.9385	0.9526	0.9391	0.8788	0.8841	0.8851	0.9814	0.9776	0.8501	0.9227

Table 4.1: wAUC for SRNet trained with cover-stego pair constraint, then refined without the pair constraint.

trained from randomly initialized weights on QF75 using standard hyper-parameters and the training schedule described in [99] with a batch size of 64. Then, it was fine-tuned on QF90 and QF95 separately for $160,000$ iterations with LR 10^{-4} for the first $100,000$ iterations, which was then divided by 2 after each $20,000$ iterations.

Another version of SRNet has also been studied in an attempt to improve the performance: refining the trained SRNet on QF75 without the cover–stego pair constraint (PC). This was done by training on each QF for $200,000$ iterations with LR 10^{-4} for $20,000$ iterations, 10^{-3} for $60,000$ iterations, and for additional $3 \times 40,000$ iterations after dividing the LR by $10, 5, 2$.

The wAUC for both SRNet versions is shown in Table 4.1 broken up by quality factors and stego methods. The improvement due to refinements is especially significant for the two larger quality factors and for UERD and J-UNIWARD. We note that all SRNet versions were trained on non-rounded $YCrCb$ values after decompressing the JPEG image. We strongly hypothesize that the improvement due to refining without pair constraint is due to restoring batch independence together with using Batch Normalization layers. Pair constraint however, helps convergence early in training from scratch. Also note that SRNet trained on all three quality factors together underperformed compared to dedicated SRNet's trained on each QF as the range is larger compared to those studied in [109].

4.3.2 ImageNet models

In Table 4.2, we show the wAUC for five different pre-trained models also broken up by quality factor and stego method. All models were refined as explained in Section 4.1.1, EfficientNet B7* was trained using pipeline (B) while the other models were trained using pipeline (A). Another round of refinement was performed with the Mish activation function [177] replacing the original Swish activation function [178]. The boost in wAUC provided by training on non-rounded pixel values consistently ranged between $0.05 - 0.1$, while the boost due to Mish activation is visualized in Figure 4.3.1. Unlike SRNet, the pre-trained models trained on all three QFs at once performed

Model	UERD			J-UNIWARD			J-MiPOD			Mixture
QF	75	90	95	75	90	95	75	90	95	
MN-xL (Mish)	0.9577	0.9675	0.9570	0.8873	0.8895	0.8919	0.9827	0.9794	0.8621	0.9322
B4 (Mish)	0.9583	0.9664	0.9538	0.8885	0.8878	0.8967	0.9828	0.9807	0.8696	0.9331
B5 (Mish)	**0.9606**	0.9691	**0.9597**	0.8911	0.8945	**0.9025**	0.9851	0.9794	0.8693	0.9360
B6 (Mish)	0.9591	0.9665	0.9567	0.8935	0.8979	0.9022	0.9842	0.9801	**0.8724**	0.9361
B7* (Mish)	0.9592	**0.9713**	0.9528	**0.9052**	**0.9112**	0.8937	**0.9876**	**0.9821**	0.8600	**0.9385**

Table 4.2: wAUC for five pre-trained models refined for steganalysis with Mish activation and on non-rounded *RGB* pixels.

similarly as models dedicated to a specific QF.

The pre-trained models offer markedly better performance than the SRNet on all quality factors and all stego methods. Also, deeper models generally achieve better detection accuracy. Figure 4.3.1 shows the performance in terms of the wAUC as a function of the model size (the number of parameters) across all stego methods and then separately for each stego algorithm. The graphs confirm that "bigger and deeper" is generally better. When viewing the performance for each stego algorithm, however, one can see that the models' accuracy varies quite a bit. For UERD, MixNet-xL reaches basically the same performance as the much bigger B6 or B7*. For J-UNIWARD and J-MiPOD, the SRNet no PC has a competitive performance and even outperforms MixNet-S and B2. The boost due to the Mish activation is mostly for J-UNIWARD and J-MiPOD. This complementary performance of the models is good as they will likely boost each other in an ensemble.

4.3.3 Pooling/stride ablation

Targeted experiments using different CNN architectures show that the resolution of the first layers is important for the final accuracy. In fact, it is well established within the steganalysis community that CNNs for steganalysis should not perform any downsampling in the first few layers [94; 99]. Figure 4.3.2 shows how models building on the ResNet stem (conv 7×7 with stride 2 followed by a 3×3 max pooling layer with stride 2) compare to the SRNet noPC as a baseline. Architectures with too much downsampling in the first layers do not follow the trend in Figure 4.3.1, and are generally weaker than the baseline. Figure 4.3.2 also shows two different trends, DenseNet and (SK-)ResNeXt families seem to perform better than ResNet and T-ResNet families. We hypothesize that this is due to the fact that DenseNet-121 and (SK-)ResNeXt-50 have a depth of 256 at the output of the first block after the stem (second highest resolution) while ResNet-34 and T-ResNet-M have a depth of 64.

Next, we compare within a single architecture (MixNet-S) how stride and max pooling in the stem

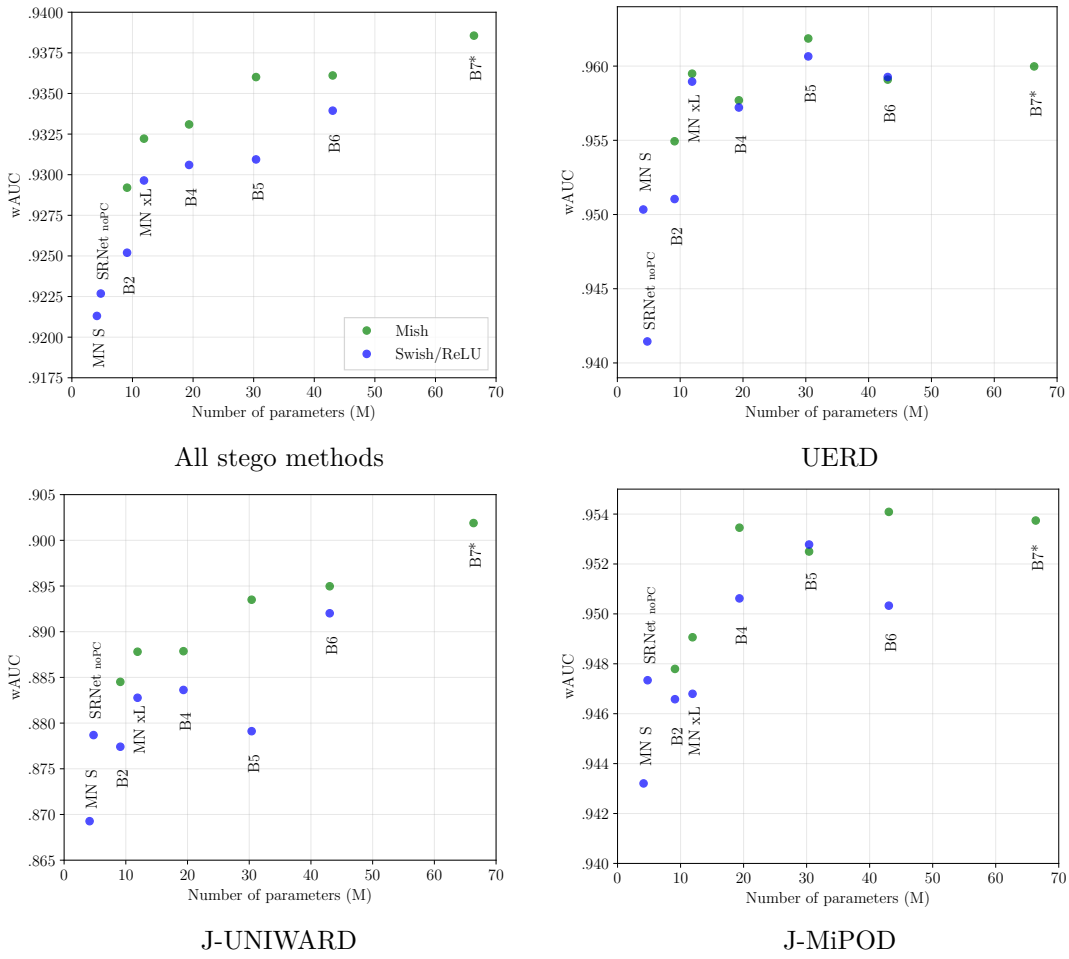


Figure 4.3.1: Performance in terms of wAUC versus model size (number of parameters) of SRNet noPC and different ImageNet pre-trained models using Swish (blue) and Mish (green) activation functions.

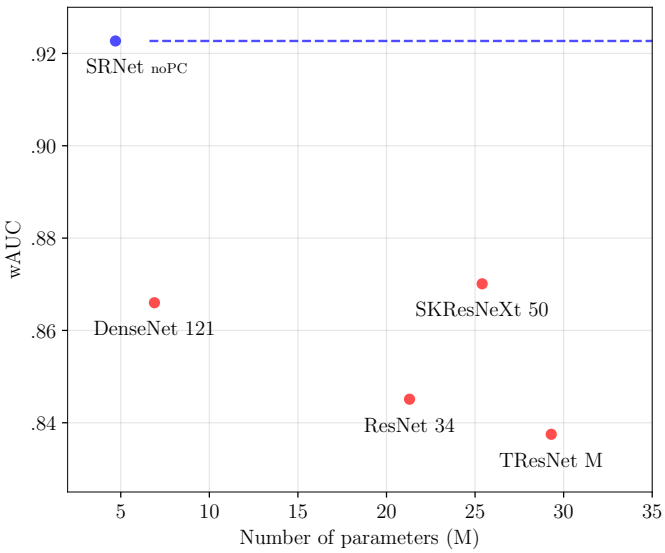


Figure 4.3.2: Performance in terms of wAUC versus model size (number of parameters) across different ImageNet pre-trained models building on the ResNet stem and SRNet noPC.

MixNet-S stem	wAUC
3×3 conv, stride 2	0.9213
3×3 conv, stride 1	0.9353
3×3 conv, stride 1$\rightarrow 3 \times 3$ avg pool, stride 2	0.8445
3×3 conv, stride 2$\rightarrow 3 \times 3$ avg pool, stride 2	0.8046

Table 4.3: wAUC for four variants of the MixNet-S architecture.

affect the performance. Table 4.3 shows the great benefit of removing the stem's stride. Table 4.3 also shows how MixNet-S performs with different stem downsampling settings when removing the stride and adding an avg. pooling layer. The performance drops considerably despite the same "resolution" as the vanilla MixNet-S stem. The drop is due to the low-pass nature of the average pooling layer, which suppresses high frequency components. The performance drops even more when keeping the strided convolution.

4.3.4 Selected case results: ALASKA I

In addition to ALASKA II experiments, we show how an ImageNet pre-trained model, EfficientNet B4 with Mish activation performs on the ALASKA I dataset [42] with nsF5, UED, EBS, and J-UNIWARD as stego schemes. The dataset preparation scripts have been modified to produce 256×256 images (tiles), embedded with double the original payload size, and compressed with JPEG quality 95. Recent work shows that CNNs trained in the spatial domain struggle to detect

	SRNet	B4 (Mish)	OneHot+SRNet
J-UNIWARD	4.55, 4.66	5.03, 5.14	4.31, 4.00
EBS	2.13, 1.20	1.44, 0.77	2.47, 1.51
UED	5.03, 5.49	5.49, 6.08	5.24, 5.63
nsF5	11.51, 24.60	13.97, 31.47	3.80, 3.14

Table 4.4: Detection error, missed detection at 5% false alarm ($P_{\mathrm{E}}, \mathrm{MD5}$) of SRNet, EfficientNet-B4 with Mish activation, and OneHot+SRNet [1] on the ALASKA I dataset (tiles, QF95, double payload) when tested against individual stego algorithms.

nsF5 for high JPEG quality factors [1]. Table 4.4 shows that the EfficientNet family also struggles with nsF5. For the other stego schemes, EfficientNet B4 (Mish) surprisingly performs similar to SRNet.

We hypothesize that ImageNet pre-trained models are more data efficient than SRNet trained from scratch – ALASKA I has twice as many images per JPEG quality factor ($25,000$ for ALASKA II and $50,000$ for ALASKA I). With the right design, ImageNet pre-trained models are able to get more reliable performance with less data, which seems to be in line with the observation made in [174]: "4.7. Accuracy benefits of ImageNet pre-training fade quickly with dataset size." Note that EfficientNet B4 (Mish) was trained using pipeline (A) described in Section 4.1.1 and initialized with ALASKA II weights. Searching for better hyper-parameters for the ALASKA I dataset might give slightly better results.

4.4 The ALASKA II Challenge

ALASKA II competitors were evaluated using the wAUC on 5,000 images split into 1,000 from a public and 4,000 from a private leader board (LB). The actual details about the split were unavailable to the teams. Each team was allowed five submissions per day consisting of a scoring of all 5,000 images, with higher score given to images more likely to be stego. The feedback about the detection accuracy was in the form of the wAUC computed only from the 1,000 images from the public LB.

The only information about the test set images that was provided was that each embedding algorithm was used with the same probability and the payload computed in the same fashion as for the training set with the average message length of 0.4 bpnzac. The images were all compressed with one of the three JPEG quality factors: 95, 90, and 75.

Model ensembles, 2nd level stacking, and final submission

Due to the competition's time constraint, and the team's late merger with Eugene Khvedchenya, our model ensemble consisted of two separate ensembles trained on different splits. Within each ensemble, we trained a 2nd-level stacking model on the detectors' outputs on the validation split (catboost [179] for Ensemble 1, xgboost [180] for Ensemble 2):

- Ensemble 1: QF (target encoded), DCTR, JRM, SRNet, MixNet-S, MixNet-xL (Mish), EfficientNet B2, B4 Mish, B5 (Mish), and B6 (Mish) (1 fold) (test score 0.9401, private score 0.931, public score 0.935)
- Ensemble 2: QF (one hot encoded), EfficientNet B6* (4 folds), B6* (Mish) (2 folds) and B7* (Mish) (2 folds) (test score 0.9424, private score 0.932, public score 0.941)

Due to the small size of the public LB, the best detector in Ensemble 1 (B6 (Mish)) performed surprisingly low on the public LB (public score 0.932), but had one of the best single model performances on the private LB (private score 0.926). We decided not to include it in our final submissions. The final blending was done by rank averaging submissions from Ensemble 1 and 2, which had a private score of 0.932 and public score of 0.944.

4.5 Conclusions

This chapter looks into the possibility to build steganalysis detectors from computer vision models pre-trained on ImageNet and refined on examples of cover and stego images. Due to time constraints, our study is limited to the setup of ALASKA II and some selected cases. Besides superior detection accuracy, the pre-trained models offer other significant advantages over models that have to be trained from scratch: the transfer learning is orders of magnitude faster than training a dedicated steganalysis CNN from scratch and is more data efficient. The refining can be done more efficiently and can be done for multiple quality factors at the same time, which drastically reduces the training complexity.

We conjecture that the superior detection performance is due to the fact that the pre-trained models have been exposed to a great variety of content and thus are able to better learn noise patterns modulated by content – the stego signal, with less data than specialized CNNs trained from scratch.

Preliminary experiments on the ALASKA I dataset show that the accuracy benefit of ImageNet pre-trained models diminishes with more training data.

This chapter poses more questions than it answers. Many interesting questions remain, such as the ability of the pre-trained models to generalize to custom JPEG quantization tables, and what the refinement should be for building detectors for spatial domain steganography.

Chapter 5

Improving EfficientNet for JPEG Steganalysis

Steganalysis with machine learning has undergone an explosive development during the past five years. This was driven by a firm belief of domain experts that the task of steganalysis is somehow fundamentally different from the main objective of computer vision, which is object classification. Fundamentally, though, detection of modern content-adaptive steganography is equivalent to detecting noise-like signals shaped by the content itself. It is thus not surprising that CNNs trained on computer vision tasks are a good starting point for transfer learning in steganalysis, as well as the closely related field of digital forensics [181; 182; 183; 184].

This is confirmed by the proliferation of pre-trained models from computer vision in the recent Kaggle competitions in Camera Model Identification,[1] Deep Fake Detection,[2] and especially, in the ALASKA II [43] JPEG steganalysis challenge.[3] Many participants in ALASKA II [2; 121; 185] used the popular EfficientNet [172] pre-trained on ImageNet [122] and refined for steganalysis in the JPEG domain. Such architectures achieved markedly better performance [2; 43] than the popular SRNet [99] considered as one of the state-of-the-art CNNs for steganalysis.

In this chapter, we investigate several "surgical modifications" of the EfficientNet family to further improve their performance for steganalysis while keeping in mind the computational complexity both in terms of FLOPs, the memory consumption, and the number of parameters. The main

[1] https://www.kaggle.com/c/sp-society-camera-model-identification
[2] https://www.kaggle.com/c/deepfake-detection-challenge
[3] https://www.kaggle.com/c/alaska2-image-steganalysis

idea for the surgical modifications follows what has already been hinted at in [93; 94] and further exploited in [99], namely that decreasing the resolution of the networks in early layers via pooling or striding negatively affects their detection accuracy as such operations enhance image content while suppressing the noise-like stego signal. Using the ALASKA II dataset as a benchmark, we investigate several types of surgical modifications in terms of their performance and computational complexity.

Note that we do not investigate after-the-fact model compression methods such as pruning [186; 187] or distillation [188], as these are conventionally applied after an initial training, and can be applied to any architecture.

We also study the EfficientNet family in other, aggressively downsampled image datasets, such as the BOSSbase, where the EfficientNet family does not seem to perform as well with respect to the SRNet. We attribute this drop to the aggressive subsampling of images and show that the proposed surgical modifications significantly improve EfficientNet detection accuracy.

In the next section, we introduce the notation used in this chapter. Section 5.2 lays out the experimental setting. Section 5.3 describes the ImageNet pre-trained CNNs and their building blocks. Section 5.4 describes the proposed "surgical modifications." Section 5.5 studies the ImageNet pre-trained EfficientNet in other datasets. The chapter is concluded in Section 5.6.

5.1 Notation

For consistency with the results from the ALASKA II competition, we evaluate the detectors' performance using the weighted area under the receiver operating characteristic (ROC) curve (wAUC). For reference, we also occasionally report the minimum average error rate under equal priors P_{E}.

FLOPs is the total number of floating point operations performed to do a single forward pass using a single image input, computed using the 'fvcore' package from Facebook Research.[4] Note that only multiplications are counted while additions as well as the bias are ignored. For example, a $k \times k \times C_{\text{out}}$ convolution layer with no stride and same padding operating on a $C_{\text{in}} \times H \times W$ has FLOPs $= k^2 HWC_{\text{in}}C_{\text{out}}$.

We measure the GPU memory needed to train a model using the peak memory consumption from the 'nvidia-smi' output. To this end, we choose a batch-size of 8, a single GPU, and the other

[4] `https://github.com/facebookresearch/fvcore`

hyper-parameters as detailed in 5.3.2. Note that the memory consumption is only an estimate and strongly depends on the implementation used. Also, it should not be confused with the total GPU memory needed since a larger batch-size is used (using data parallelism over multiple GPUs) as detailed in 5.3.2.

5.2 Experimental setting

The ALASKA II [43] dataset contains 3×25,000 different cover images compressed with quality factors 75, 90, and 95, and the same number of stego images embedded with J-UNIWARD [19], J-MiPOD [176], and UERD [21], making the training set size 4×75,000 images. The payload embedded in each image was scaled so that all images were approximately equally difficult to detect – comparatively smaller payloads were embedded in smooth images with larger payloads in highly textured or noisy images. The average payload embedded across the database was 0.4 bits per non-zero AC DCT coefficient (bpnzac). The dataset was randomly divided into three sets with 4×3×22,000, 4×3×1,000, and 4×3×2,000 images, for training, validation, and testing respectively. The splits were made compatible with those used in [2]. Note that, unlike [2], this chapter does not report results using test-time augmentation (TTA).

Section 5.5.1 describes additional datasets used to investigate the ImageNet pre-trained models in different settings. We describe those settings within Section 5.5 for better readability.

5.3 EfficientNet for JPEG steganalysis

5.3.1 EfficientNet and SE-ResNet

Our investigation is constrained to the EfficientNet family widely used by the top competitors in the Alaska II challenge, and ultimately compared with the SE-ResNet [189] used by the winner of the competition. Other work [2] reports on the MixNet [173] architecture, which we argue achieves a very similar performance as the EfficientNet due to strong architecture similarities. Thus, we do not report results using the MixNet family for this reason but expect our contributions to transfer to such similar architectures.

The EfficientNet is built using the Inverted Residual Block [190] (IR) depicted in Figure 5.3.1. The lightweight depth-wise separable convolution (D-Conv) is the key to the network's efficiency. On

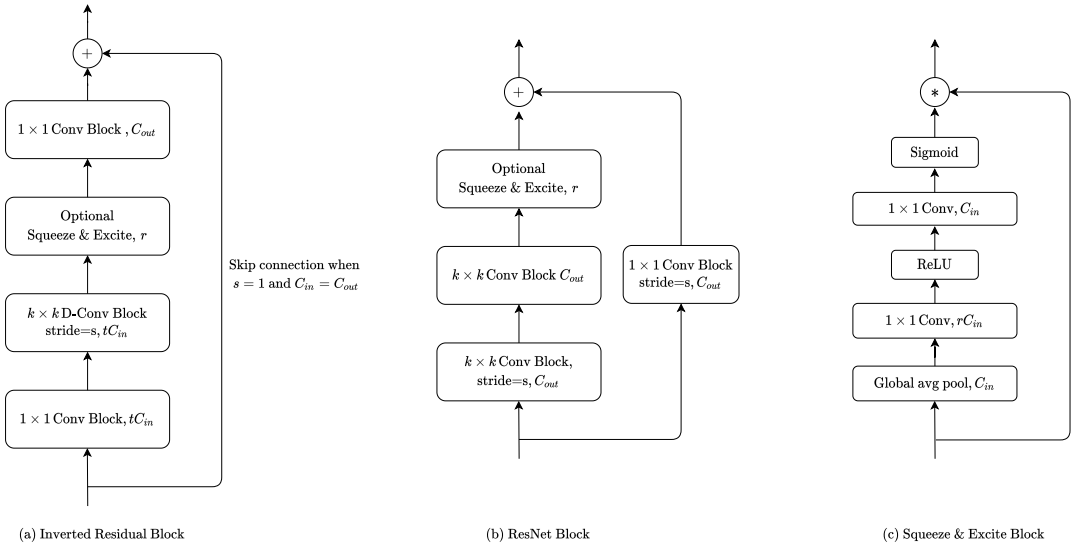


Figure 5.3.1: (a) The Inverted Residual block used in the EfficientNet architecture, t denotes the expansion parameter of the block. (b) The ResNet Block used in SE-ResNet18. (c) The Squeeze & Excite block, r denotes the reduction parameter. In (a) and (b) Conv Blocks are composed of a Convolution layer, Batch Normalization layer, and an Activation.

the other hand, the SE-ResNet uses the classical ResNet Block [165]. Both architectures use the Squeeze & Excite Block [189] with different reduction parameters.

5.3.2 Transfer learning procedure

Fine-tuning ImageNet pre-trained models on a steganalysis task is done using multi-class cross-entropy loss and the AdamW optimizer with 10^{-2} weight decay, for 60 epochs using a cosine learning rate scheduler with a start LR of 10^{-3} and end LR of 2×10^{-5}, and D_4 training augmentation. The model is converted and trained in Automatic Mixed Precision (AMP). We used a minimum batch size of 24, which was increased for smaller architectures to speed up training. The mini-batches were not pair-constrained, which means that on average, one batch included 1/4 cover images and 3/4 stego images randomly sampled. The JPEG images were decompressed to RGB color space without rounding or clipping. After training, we chose the best checkpoint based on the wAUC metric on the validation set.

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
Vanilla B0	.92601	2.15	4.01	3,354
B0 no stride L0	.92844	8.31	4.01	10,162
B0 no stride L2	.93552	30.73	4.01	24,184
B4 no stride L0	.94408	31.63	17.56	21,484
B6 no stride L0	.94618	69.97	40.74	40,186

Table 5.1: wAUC, FLOPs, and the number of parameters of different modifications of the EfficientNet family.

5.4 "Surgical modifications" improve EfficientNet

The Alaska II Kaggle competition has shown that many successful ImageNet pre-trained CNNs can achieve a very competitive performance when fine-tuned on a steganalysis task. Moreover, carefully modified ImageNet pre-trained CNNs can be made even better. We consider two types of "surgical modifications": ablation of downsampling elements in the architectures (stride, pooling) and insertion of additional layers operating on high-resolution feature representations.

5.4.1 Stem stride ablation

In [2], the authors show that the performance of the MixNet-S can be significantly improved by removing the stride from its stem. This was also reported by many competitors.[5] We report similar results with the EfficientNet family for consistency with the chapter's experiments. Table 5.1 shows that removing strides from the stem (no stride L0) and from the next strided block (no stride L2) of EfficientNet B0 significantly improves the performance but comes at a substantial price in FLOPs and memory requirements because the convolutions will operate on 4× larger volumes after each removed stride. Removing strides from the stem of EfficientNet B4 gives a stellar performance but with an unreasonably high memory consumption making the training extremely slow.

For SE-ResNet, we first remove the max-pooling in the stem and train for 10 epochs, then remove the stride and continue the training. Note that this was not described by the winner of the Alaska II competition in [185] but was essential to successfully train the modified architecture with the hyper-parameters of our experiments. Similarly, this surgical modification comes at a substantial price in FLOPs.

Note that disabling stride and/or pooling in the stem does not change the number of parameters

[5] `https://www.kaggle.com/c/alaska2-image-steganalysis/discussion`

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
Vanilla SE-ResNet	.87661	9.53	11.26	3,084
SE-ResNet no stride/pool L0	.94231	144.89	11.26	8,468

Table 5.2: wAUC, FLOPs, and the number of parameters of the SE-ResNet-18 and its modified version.

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
Vanilla B0	.92601	2.15	4.01	3,354
B0 - original pre-stem	.92902	7.16	4.04	3,978
B0 - pre-stem IR blocks	.93300	4.98	4.03	6,460
B0 - post-stem IR blocks	.93313	4.88	4.02	6,812

Table 5.3: wAUC, FLOPs, and the number of parameters of different modifications of the EfficientNet-B0.

of a network, however the computational cost (GPU memory and FLOPs) to train it increase significantly. This motivates our choice to use FLOPs as a model-complexity measure.

5.4.2 Unpooled layers implant

In this section, we show that the performance can be improved at a much lower cost in term of FLOPs and memory requirements by inserting layers at influential parts of the architecture, namely in early layers to mimic the "unpooled layers" of steganalysis CNNs, such as the SRNet.

Some Alaska II competitors reported performance increase when adding layers to the EfficientNet architecture [191]. We call this surgical modification "pre-stem insertion," since the layers are implanted before the stem. The original modification described in [191] only included 3 convolutional blocks with an increased number of channels. We show that using 4 blocks and a large number of channels from the first block is beneficial. Note that we did not modify the last two layers as described in [191] as this degraded the performance in our experiments.

Additionally, we introduce a new surgical modification called "post-stem" insertion, which (i) disables the downsampling operation in the stem and (ii) inserts convolutional blocks after the stem, last of which has a stride of 2. In essence, post-stem and pre-stem are very similar architectures, with post-stem being more computationally efficient and more accurate as shown in Table 5.3.

We choose to study the post-stem surgery further by changing some of its design hyper-parameters. Increasing the expansion parameter t of the implanted Inverted Residual blocks improves the repre-

Surgical modification	wAUC	FLOPs (B)	Params (M)	Mem (MiB)
B0 - post-stem IR blocks $t = 1$	.93313	4.88	4.02	6,812
B0 - post-stem IR blocks $t = 4$	.93506	12.15	4.05	13,840
B0 - post-stem ResNet blocks	.93623	18.31	4.08	6,488

Table 5.4: wAUC, FLOPs, and the number of parameters of different variants of the post-stem surgical modification with the EfficientNet-B0.

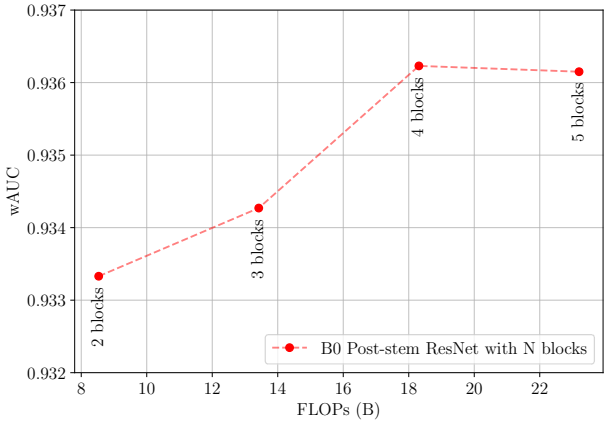


Figure 5.4.1: wAUC vs. FLOPs of EfficientNet-B0 with a post-stem modification and a varying number of inserted convolutional blocks.

sentation capacity of those layers by allowing to form a larger number of noise residuals. Using the ResNet blocks instead of the lightweight Inverted Residual blocks also allows the implants to form more complex residuals. Both changes improve the performance as shown in Table 5.4. These improvements prove, yet again, the importance of the early unpooled layers in the architectures. Note that these improvements also come at a FLOPs cost, making the surgically modified model more computationally demanding but still having a better performance-memory to compute trade-off than the stride ablation studied in Section 5.4.1.

We also study the effects of the number of inserted layers in Figure 5.4.1, which shows that the performance increases with increasing number of blocks, then saturates at 4 inserted ResNet blocks. He hypothesize that this optimal number of inserted layers depends on the cover and/or stego source and would require to be validated accordingly. However, for the sake of the experiments in this chapter, we fix a number of inserted layers of 4 for the post-stem surgery, keeping in mind that this number might be different for different scenarios (e.g. spatial domain steganalysis).

Figure 5.4.2 describes all insertion strategies proposed and studied in this chapter. The insertion strategies are shown with 4 added blocks, determined by the results shown in Figure 5.4.1. We show

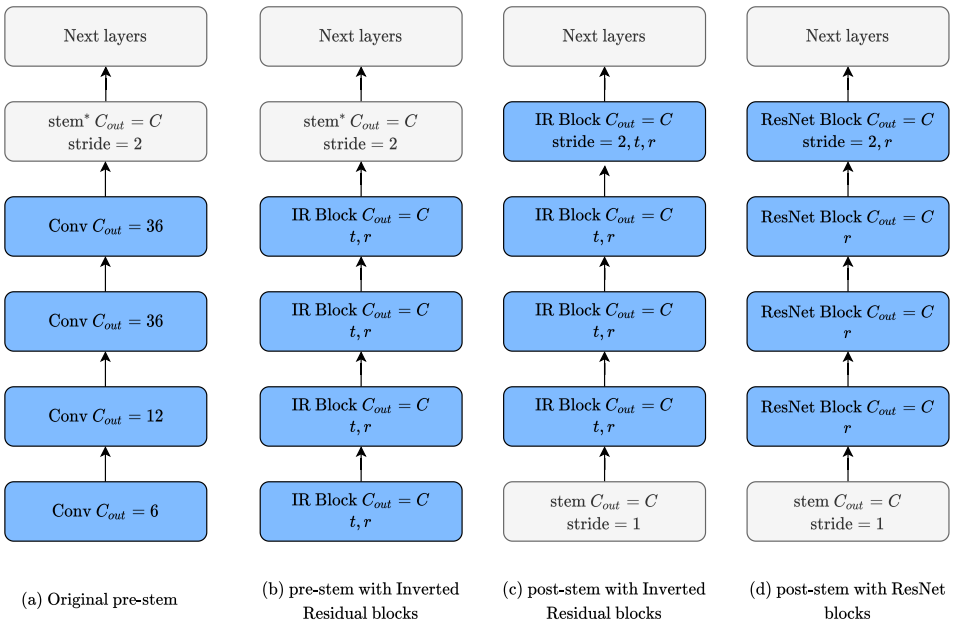


Figure 5.4.2: All surgical modifications studied. Blue blocks are the inserted blocks. In (a) and (b) the stem* convolutional kernels are duplicated and concatenated to match the previous layer's dimension.

the performance of the modifications studied in Figure 5.4.3, removing the stride performs best but at a significant memory cost as discussed in Section 5.4.1, the next best modification coming at a lower memory cost is the post-stem with ResNet blocks. The SE-ResNet18 with stride and pooling removed has a significant FLOPs count due to the use of the expensive ResNet blocks, but has a reasonable memory footprint thanks to its reduced size. A similar trend is observed with P_{E} as a metric instead of the wAUC. The equivalent of Figure 5.4.3 with the P_{E} metric is shown in Figure **??** in the Appendix for reference.

5.4.3 Do we gain from ImageNet pre-training of modified CNNs?

A relevant question regarding the surgical modifications is whether we might gain from pre-training the modified architectures on ImageNet instead of only modifying them at the transfer learning stage. This question is especially relevant for the post-stem modification where randomly initialized layers are inserted within a network already trained on ImageNet. Table 5.5 shows that there is no substantial benefit from training the modified EfficientNet on ImageNet. The table also includes a "control" vanilla EfficientNet-B0 pre-training to show that our local ImageNet pre-training matches

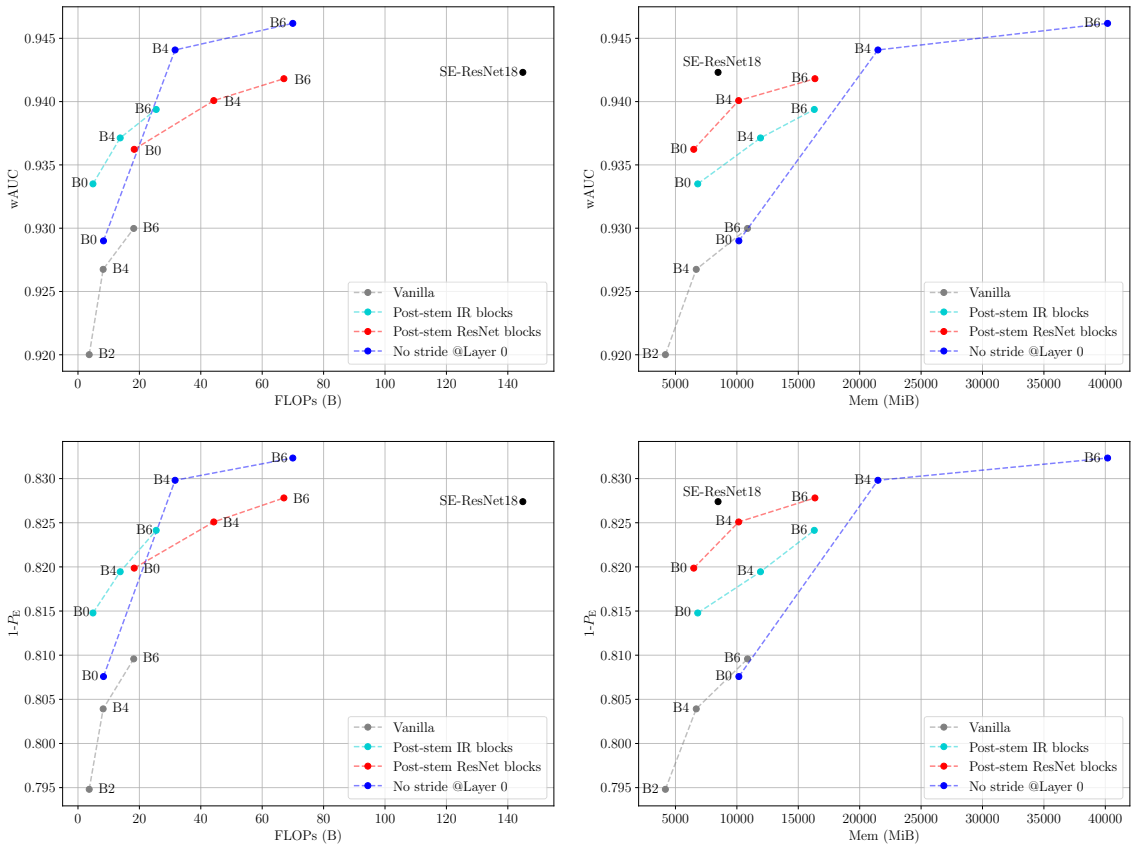


Figure 5.4.3: wAUC (top) and $1 - P_{\mathrm{E}}$ (bottom) vs. FLOPs and memory requirements of different surgical modifications. The EfficientNet B6 with stride disabled and B6 post-stem with ResNet blocks achieve a comparable performance to the SE-ResNet18 with pool and stride disabled, with one half of the FLOPs. For reference, we include the vanilla versions of the EfficientNet trained using the schedule described in [2] in Section II.A.

	Pre-trained w/ modification	Surgically modified
B0 - post-stem	.93391	.93313
(control) Vanilla B0	.92609	.92601

Table 5.5: wAUC of EfficientNet-B0 with post-stem with Inverted Residual Block $t = 1$ and unchanged. Modification done at the pre-training stage or at the transfer learning stage.

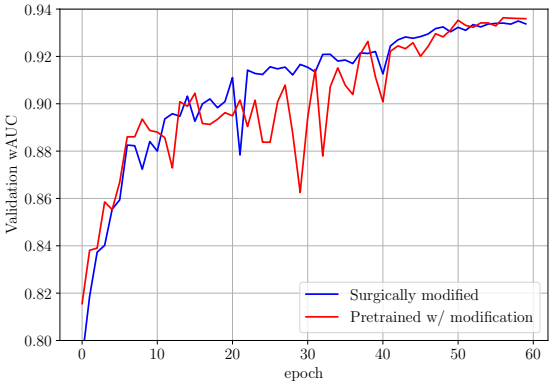


Figure 5.4.4: Validation wAUC at different training epochs of the EfficientNet-B0 with post-stem with Inverted Residual Block $t = 1$ both pre-trained with the modification or surgically modified.

the one done by the EfficientNet-pytorch[6] library in terms of performance in the downstream task. This means that it is safe to "surgically" apply the modifications at the transfer learning stage. Pre-training on the ImageNet dataset was done for 100 epochs with the SGD optimizer with 0.9 momentum, a weight-decay of 10^{-5}, the OneCycle learning rate scheduler with a maximum learning rate of 0.5, a minimum of 10^{-3}, and a batch-size of 512.

Figure 5.4.4 shows the validation wAUC at different training epochs of the two versions of the B0 - post-stem in Table 5.5. We note that while the surgically modified network starts lower than the the one pre-trained with the modification, the two versions eventually converge to a very close peak performance as shown in Table 5.5. The early epochs of post-stem surgically modified networks usually exhibit a lower performance due to the randomly initialized convolutional blocks inserted, but the performance increases given enough epochs.

[6] `https://github.com/lukemelas/EfficientNet-PyTorch`

5.5 Where does EfficientNet shine?

5.5.1 Additional experimental setting

For this section, we use two more datasets to compare the effectiveness of the ImageNet pre-trained EfficientNet in different settings.

BOSSbase+BOWS2 is the union of BOSSbase 1.01 [143] and BOWS2 [144] converted to grayscale and resized to 256×256 using Matlab's 'imresize' with default parameters. We use JPEG quality factors 75, 90, and 95 and embedding schemes J-UNIWARD, J-MiPOD, and UERD at 0.4, 0.4 and 0.2 bnzac respectively (fixed payload sender). The dataset is randomly divided into three sets with $4\times3\times14,000$ (BOSSbase+BOWS2), $4\times3\times1,000$ (BOSSbase), $4\times3\times5,000$ images (BOSSbase) for training, validation, and testing respectively. The splits are also made compatible with [99]. Note that for consistency with the results from the ALASKA II competition, we used the same versions of the previously listed embedding schemes as in the competition's dataset. New versions of the J-MiPOD [192] or correctly implemented UERD were not considered.

We create a new dataset called ALASKA II BOSS-style which contains raw images from the ALASKA II dataset, processed using the BOSSbase processing script and resized to 256×256 using ImageMagick's resize with default parameters. We use the same embedding script as the ALASKA II dataset with an average payload across database of 0.2 bpnzac (Detectability Limited Sender [193]). The dataset is divided into the same splits as the ALASKA II dataset.

5.5.2 Training and transfer learning procedure

We use the SRNet [99] without the pair-constraint [2] as a baseline to evaluate EfficientNet in these additional datasets. SRNet was first trained on QF75 with the pair-constraint for 200 epochs then refined on QF75, 90, and 95 without the pair-constraint for another 100 epochs. Training was done with the multi-class cross-entropy loss, using the Adamax optimizer with a 10^{-4} weight decay, the OneCycle learning rate scheduler with a start LR of 4×10^{-5}, a maximum LR of 10^{-3}, and an end LR of 2×10^{-5}. Inputs were also transformed to RGB for color images without rounding or clipping, and divided by 255 before feeding to the network.

For the ImageNet pre-trained EfficientNet and SE-ResNet18, we used the same hyper-parameters as in 5.3.2. For grayscale BOSSbase+BOWS2, we insert a 1×1 convolution layer with 3 output channels before the stem to match the input channels of the pre-trained stem.

Dataset	ALASKA II			BOSSbase+BOWS2			ALASKA II BOSS-style		
Sender	DeLS 0.4 J-UNI	DeLS 0.4 J-MiPOD	DeLS 0.4 UERD	PLS 0.4 J-UNI	PLS 0.4 J-MiPOD	PLS 0.2 UERD	DeLS 0.2 J-UNI	DeLS 0.2 J-MiPOD	DeLS 0.2 UERD
SRNet	.8718	.9427	.9364	.9634	.9780	.9773	.9903	.9142	.9279
B4	.8783	.9475	.9540	.9528	.9723	.9812	.9892	.9021	.9436
B4 no stride L0	**.9092**	.9592	**.9636**	.9700	.9828	.9857	.9925	.9302	.9528
B4 post-stem ResNet blocks	.9006	.9574	.9620	**.9746**	.9852	**.9879**	.9923	.9302	**.9532**
B6 post-stem ResNet blocks	.9057	.9566	.9629	.9731	**.9880**	.9842	**.9927**	**.9382**	.9503
SE-ResNet18 no stride/pool L0	.9045	**.9610**	.9611	.9729	.9839	.9865	.9911	.9234	.9491

Table 5.6: wAUC of different modifications of the EfficientNet, the SE-ResNet18, and the SRNet as a baseline in three different datasets with their respective sender strategies.

5.5.3 The effect of BOSS-style processing

This investigation section started by noticing the surprising difference between the first two columns in Table 5.6: in the ALASKA II dataset, the vanilla EfficientNet-B4 outperforms SRNet, but in the BOSSbase+BOWS2 dataset, SRNet outperforms the EfficientNet-B4. Note that this was also observed in [2] with the ALASKA I [42] dataset.

In [2], the authors hypothesized that this shift is due to the fact that ImageNet pre-trained models might be more data efficient than the SRNet trained from scratch.

However, we show that the main differences between the first two columns of Table 5.6 are mostly due to the cover processing pipeline because using a "BOSSbase-style" ALASKA II processed database again shows EfficientNet-B4 underperforming.

Table 5.6 shows that vanilla versions of ImageNet pre-trained models underperform in strongly subsampled cover sources, such as BOSSbase+BOWS2 and BOSS-style datasets. We hypothesize that this is due to their lack of unpooled layers. Strongly subsampled cover sources exhibit more high frequency content, which will require more layers operating at the original resolution. The proposed surgical modifications help mitigate this effect as they introduce more unpooled layers into the architecture.

Note that this observation does not hold for the UERD embedding scheme. This is because the ALASKA II competition used a faulty implementation of UERD that concentrates most of the embedding changes at the image boundary and thus is much less dependent on the cover source.

We verify that BOSS-style datasets have indeed more high frequency content by computing the average energy of the KB [9] residual over 500 images from the original ALAKA II and the ALASKA II BOSS-style. Figure 5.5.1 shows the distributions of the square root of the average energy (the square root helps avoid large outliers) for the two datasets in the uncompressed format, JPEG compressed with qualities 95, 90, and 75. Figure 5.5.1 shows that indeed, BOSS-style processing

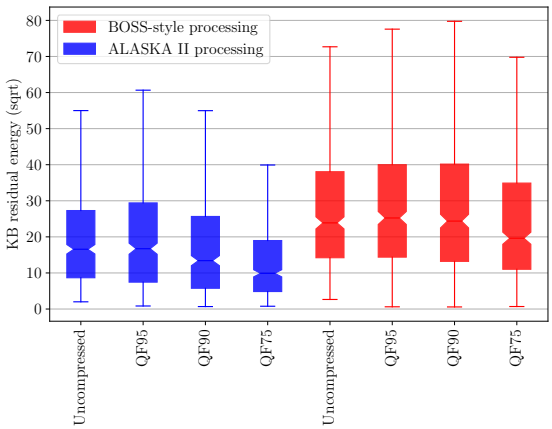


Figure 5.5.1: Distribution of the square root of the average KB residual energy across 500 images uncompressed and JPEG compressed with qualities 95, 90, and 75 from the ALASKA II and the ALASKA II BOSS-style datasets.

contains complex content when measured by the inability to predict the pixel values from their neighborhoods using the KB filter.

Table 5.6 also shows that the EfficientNet B6 (amd B4) post-stem with ResNet blocks have a very comparable performance to the SE-ResNet18 no stride/pool L0 (even slightly better) on the BOSSbase+BOWS2 and ALASKA II BOSS-style datasets. Note that on these datasets, the post-stem surgery performs better than the stride ablation, unlike in the ALASKA II dataset as shown in Figure 5.4.3.

5.6 Conclusions

We propose and study several different ways to modify the EfficientNet architecture to significantly improve performance for JPEG steganalysis. These so called "surgical modifications" are done at the transfer-learning stage to substantially improve the performance upon the original (vanilla) EfficientNet architectures. The post-stem modification boosts the performance while keeping the computational cost and the memory requirements reasonable by increasing the number of unpooled layers in the architecture. Removing the stride in the stem of the EfficientNet architectures, on the other hand, requires large GPU memory for training. The modified models reach state-of-the-art performance with less than 1/2 of the FLOPs of the current best model on the ALASKA II dataset.

We also test the EfficientNet family in different datasets and notice that in strongly subsampled cover sources (e. g., BOSSbase+BOWS2), they underperform with respect to the SRNet due to

Backbone	Model Surgical Modification	UERD QF75	 QF90	 QF95	J-UNIWARD QF75	 QF90	 QF95	J-MiPOD QF75	 QF90	 QF95	Mixture	FLOPs (B)	Params (M)	Mem (MiB)
B0	None	.9513	.9653	.9517	.8766	.8777	.8847	.9820	.9740	.8532	.92601	2.15	4.01	3,354
	No stride L0	.9529	.9640	.9539	.8753	.8830	.8881	.9811	.9793	.8630	.92844	8.31	4.01	10,162
	No stride L2	.9539	.9637	.9550	.8892	.8944	.9054	**.9856**	.9839	.8779	.93552	30.73	4.01	24,184
	Original pre-stem	.9515	.9623	.9481	.8768	.8846	.8941	.9819	.9785	.8701	.92902	7.16	4.04	3,978
	Pre-stem IR blocks $t = 1$	.9552	**.9662**	.9553	.8866	.8915	.8967	.9840	.9792	.8686	.93300	4.98	4.03	6,460
	Post-stem IR blocks $t = 1$	.9532	.9649	.9563	.8854	.8925	.9003	.9818	.9794	.8735	.93313	4.88	4.02	6,812
	Post-stem IR blocks $t = 4$	.9523	.9640	**.9569**	.8904	.8973	.9048	.9834	.9815	.8738	.93506	12.15	4.05	13,840
	Post-stem ResNet blocks	**.9552**	.9650	.9559	**.8932**	**.9004**	**.9066**	.9830	**.9818**	**.8746**	**.93623**	18.31	4.08	6,488
B4	None	.9542	.9624	.9497	.8783	.8788	.8860	.9805	.9759	.8596	.92675	8.20	17.56	6,692
	No stride L0	.9608	**.9699**	**.9631**	**.9069**	**.9116**	**.9149**	**.9879**	**.9851**	**.8853**	**.94408**	31.63	17.56	21,484
	Post-stem IR $t = 1$	.9587	.9671	.9578	.8956	.8996	.9067	.9810	.9790	.8773	.93713	13.75	17.57	11,924
	Post-stem ResNet blocks	**.9620**	.9692	.9585	.8971	.9037	.9097	.9858	.9822	.8821	.94008	44.23	17.72	10,146
B6	None	.9534	.9644	.9514	.8848	.8901	.8920	.9808	.9764	.8601	.92998	18.16	40.74	10,868
	No stride L0	**.9625**	**.9715**	**.9628**	**.9093**	**.9139**	**.9168**	**.9912**	**.9871**	**.8881**	**.94618**	69.97	40.74	40,186
	Post-stem IR $t = 1$	.9590	.9685	.9542	.8986	.9036	.9072	.9850	.9830	.8841	.93938	25.48	40.77	16,304
	Post-stem ResNet blocks	**.9625**	.9707	.9597	.9054	.9073	.9138	.9836	.9813	.8817	.94181	67.07	40.98	16,356
SE-ResNet18	None	.9321	.9363	.9298	.8019	.7668	.7534	.9687	.9549	.7615	.87661	9.53	11.26	3,084
	No stride/pool L0	**.9621**	**.9686**	**.9570**	**.8999**	**.9104**	**.9116**	**.9864**	**.9853**	**.8881**	**.94231**	144.89	11.26	8,468

Table 5.7: wAUC, FLOPs, memory, and the number of parameters of different architectures and surgical modifications in the ALASKA II dataset.

their lack of unpooled layers. The proposed surgically modified EfficientNet architectures overcome this issue and surpass the popular SRNet on a variety of datasets.

More broadly, this chapter confirms that off-the-shelf successful computer vision architectures, such as the EfficientNet, can reach unparalleled performance in JPEG steganalysis. No special elements were added to the architecture besides the unpooled layers known to be beneficial for steganalysis.

5.7 Numerical results

We show the details of the performance of all architectures studied in this chapter for every stego scheme and JPEG quality factor in the ALASKA II dataset in Table 5.7.

Chapter 6

CNN Steganalyzers Leverage Local Embedding

Recently, steganalysis has undergone an explosive development due to employment of deep convolutional neural networks (CNNs) [146]. Major improvements in detection accuracy have been achieved for all embedding algorithms and both domains. Immediately, speculations appeared about why these detectors perform so much better than classifiers trained on high-dimensional rich media models. The usual explanation is the network's ability to jointly optimize the image representation ("feature formation") as well as the classifier. Indeed, to keep the dimensionality of co-occurrences from which rich models are built reasonably low, noise residuals need to be harshly truncated and quantized. Furthermore, training on large datasets becomes computationally infeasible even with low-complexity classifiers [73; 70].

There is one more fundamental difference between CNN detectors and rich models. The latter are by their construction macroscopic quantities of local statistics collected in a global fashion from the entire image. Rich models are essentially collections of histograms. This limits them to being predominantly "integrators" of local embedding traces across the image that achieve non-trivial detection power by leveraging some form of the Central Limit Theorem (CLT). In contrast, CNNs do not natively form histograms, and instead process the outputs of convolutions or "noise residuals" in a different fashion that is believed to allow both integration as well as detection of localized embedding traces. To the best of our knowledge, however, no study has been put forward that would present evidence for this claim.

Aided with visualization tools, we argue that CNN detectors leverage Locally Detectable Embedding Artifacts (LDEAs) in their decision making. Leaving a few stego blocks with LDEAs in the stego image is enough for a reliable detection even with other CNN architectures. In contrast, rich models are not necessarily able to correctly classify all stego images with LDEAs. By taking a closer look at modern content-adaptive algorithms J-MiPOD [176] and J-UNIWARD [19], and older embedding schemes (F5 [117], –F5 [194], and Jsteg), we discover that LDEAs are mostly associated with content-creating changes when the magnitude of a DCT coefficient is increased and, especially when a high-frequency cover DCT equal to 0 is changed to a non-zero value. Additionally, we argue that LDEAs and inhibition play a role when training a multi-class detector to distinguish between selection channels of different embedding algorithms. Our findings provide valuable qualitative and human interpretable feedback to the steganographer that could be taken into consideration for design of future stego algorithms.

In the next section, we describe the datasets and detectors employed in our experiments. Section 6.2 describes visualization tools used in this work. LDEAs are defined in Section 6.3, which contains case studies involving JPEG steganographic algorithms. In Section 6.4, we study a multi-class CNN distinguishing between bUERD, J-UNIWARD, and covers to demonstrate that it uses inhibitory response with LDEAs on the image boundary. Section 7.7 concludes the chapter.

6.1 Experimental Setting

6.1.1 Datasets and detectors

We use the ALASKA II 256×256 dataset [43], which contains $3\times25{,}000$ cover images compressed with quality factors 75, 90, and 95. The covers were randomly divided into three sets with $3\times22{,}000$, $3\times1{,}000$, and $3\times2{,}000$ images for training, validation, and testing, respectively. The images were embedded only in the luminance channel Y. The findings of this chapter are consistent when using the 256×256 grayscale BOSSbase+BOWS2 cover dataset but we do not report on them due to space constraint.

EfficientNet B4 [172] was pre-trained on ImageNet [122] and refined for steganalysis in the JPEG domain [2; 195] with the same training schedule as in Section 4.2 in [195]. No modifications were done to the EfficientNet B4 architecture besides changing the original Fully Connected (FC) layer to a binary classification FC or a three-class FC in Section 6.4. We also use the SRNet [99] trained

without pair constraint as in [2] and DCTR [81] with FLD ensemble [70].

6.2 Toolbox

6.2.1 Integrated Gradients

Integrated Gradient (IG) [196] is a technique for computing a map describing the importance of each pixel when facing a stego image. The soft output of a CNN is a function $f : \mathbb{R}^N \to \mathbb{R}$, $N = 256 \times 256$, whose domain are 256×256 images. The cover, stego, and the baseline image are denoted, respectively, $\mathbf{c}$, $\mathbf{y}$, and $\mathbf{b}$. The IG algorithm is a pixel attribution function:

$$\phi(f, \mathbf{y}, \mathbf{b}) = (\mathbf{y} - \mathbf{b}) \odot \int_0^1 \frac{\mathrm{d}f\left(\mathbf{b} + \alpha(\mathbf{y} - \mathbf{b})\right)}{\mathrm{d}\mathbf{y}} \, \mathrm{d}\alpha, \tag{6.2.1}$$

where $\mathrm{d}f/\mathrm{d}\mathbf{y} \in \mathbb{R}^N$ is the gradient of f w.r.t. to the input $\mathbf{y}$, $\odot$ denotes element-wise multiplication, and $\phi \in \mathbb{R}^N$. This algorithm belongs to path methods [197] and satisfies some desirable properties, such as, but not limited to, linearity, symmetry preserving, and completeness $\sum_{p=1}^{N} \phi_p(f, \mathbf{y}, \mathbf{b}) = f(\mathbf{y}) - f(\mathbf{b})$. It accumulates the gradients on convex combinations of the baseline $\mathbf{b}$ and the input s. This accumulation encapsulates how the network's output evolves from $f(b)$ to $f(s)$. The multiplication by $s - b$ comes from the fact that the derivative is taken with respect to the path $\gamma(\alpha) = \mathbf{b} + \alpha(\mathbf{y} - \mathbf{b})$. In practice, this multiplication can be omitted as we do in Section 6.4. The choice of b will be discussed in Section 6.2.1.1. The integral is approximated using a Riemman sum with 100 steps and the gradient is evaluated using pytorch's automatic differentiation. We use the implementation available in the Captum library.[1]

The map $\phi(f, \mathbf{y}, \mathbf{b})$, which has the same shape as the input of the CNN, 256×256, is then averaged over 8×8 non-overlapping blocks along the spatial dimensions to obtain a $256/8 \times 256/8$ block importance map $\psi_r(f, \mathbf{y}, \mathbf{b})$, $r = 1, \ldots, 32 \times 32$.

6.2.1.1 Choice of the baseline: Top k insertion test

While the IG algorithm can use an arbitrary baseline, the cover version of the stego image is the most appropriate baseline because it relates to the concept of missingness [198]. The cover represents exactly the missing signal of interest, the stego noise. Note that when using the cover image c as

[1] `https://github.com/pytorch/captum`

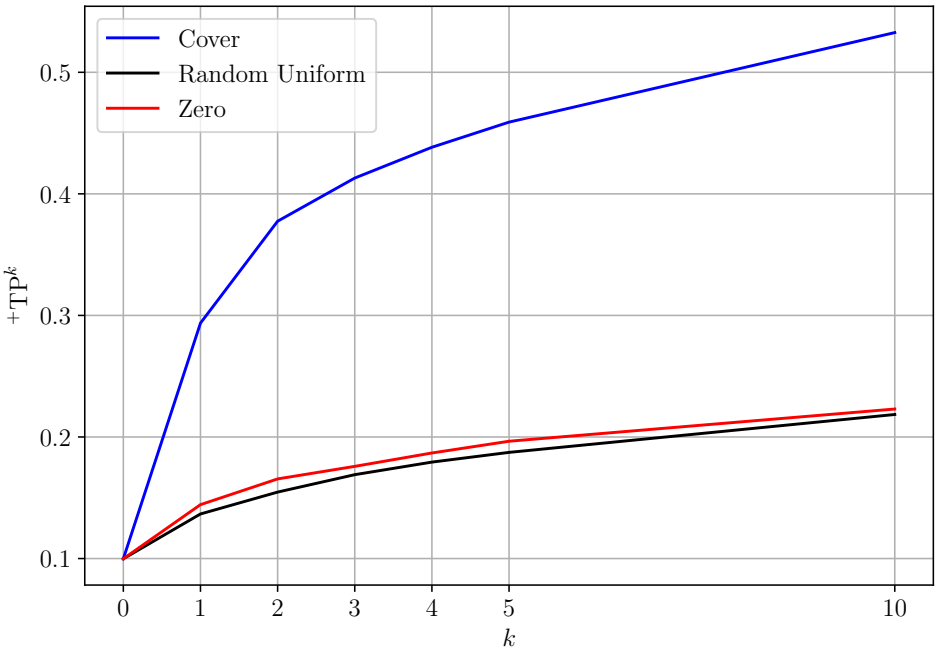


Figure 6.2.1: $^{+}\text{TP}^{k}$ rate for covers $^{+}\mathbf{c}^{k}$ with top k inserted stego blocks determined using IG with different baselines: cover, random uniform, and zero. EfficientNet B4 trained for 0.5 bpnzac J-MiPOD.

a baseline, the difference from the baseline in the IG algorithm are the stego changes $\mathbf{y} - \mathbf{c}$ in the spatial domain. Since the cumulative gradients (6.2.1) are modulated by the stego changes, $\phi(f, \mathbf{y}, \mathbf{c})$ and $\psi(f, \mathbf{y}, \mathbf{c})$ are zero for pixels and blocks without any changes.

We now compare three choices for the baseline to show that the cover baseline is a suitable choice: cover, zero, and a random image with each pixel sampled independently from a uniform distribution on $[0, 1)$. For each cover-stego $(\mathbf{c}, \mathbf{y})$ pair from the test set, we compute $\psi(f, \mathbf{y}, \mathbf{c})$ and identify "top" k blocks with the largest ψ that contain at least one stego change. Then, we generate from the cover $\mathbf{c}$ a new "stego" image, $^{+}\mathbf{c}^{k}$, by only keeping the embedding changes in the top k blocks (blocks with maximal attribution ψ). The $^{+}\text{TP}^{k}$ rate is the percentage of $^{+}\mathbf{c}^{k}$ images in the test set predicted as stego using a decision threshold set for 10% False Alarm (FA) rate. Figure 6.2.1 shows the $^{+}\text{TP}^{k}$ for EfficientNet B4 trained on 0.5 bpnzac J-MiPOD and three types of baseline images as a function of k. The cover image is clearly the best baseline for identifying the blocks that most increase the confidence of the network.

Note that even though $^{+}\mathbf{c}^{k}$ are not necessarily samples of stego images (even with a lower payload), they are natural looking images, unlike insertion/ablation evaluations done in the explainable ML literature (c.f. [198]), where insertion/ablation tests are done by blurring/dropping areas of the

image. This makes the inputs used in the top k insertion test fairly close to the original training distribution.

6.2.2 Last activation

In addition to IG, we use a gradient-free localization technique which we call "last activation." We essentially disable the last global pooling of a CNN and use the FC layer weights and bias as a 1×1 convolution to obtain a 16×16 matrix for the SRNet and 8×8 matrix for the EfficientNet B4 (for an input of shape 256×256). Then, we only keep the positive values in this matrix (positive logits of the stego class) and nullify the rest (rectification) to obtain a visualizable activation map. For example, Figure 6.2.2 shows the last activation of EfficientNet B4 for a J-MiPOD and J-UNIWARD image and the corresponding IG block importance maps. The figure will be commented upon in more detail in Section 6.3.4.

6.3 Locally Detectable Embedding Artifacts LDEAs

In this section, we define the concept of a Locally Detectable Embedding Artifact (LDEA) and use the tools explained above to analyze how CNNs detect selected modern content-adaptive and old steganographic methods. Three modern stego methods are included in the study: J-UNIWARD [19], J-MiPOD, and bUERD [21]. The last is a version of the UERD algorithm as implemented during ALASKA II. Among older embedding paradigms, we selected F5 [117], $-$F5 [194] which reverses the embedding operation of F5 to increasing the absolute value of DCT coefficients instead of decreasing as in F5, and Jsteg [199]. For modern stego schemes, the payload was fixed at 0.5 bpnzac, while for older schemes it was scaled down to avoid perfect detection by modern steganalyzers. The relative payloads α (in bpnzac) for $-$F5 and Jsteg were set to induce the same number of embedding changes $m = N_{0\mathrm{AC}} H_2^{-1}(\alpha_{-\mathrm{F5}})$ which happens when $\alpha_{\mathrm{Jsteg}} = 2H_2^{-1}(\alpha_{-\mathrm{F5}})$, where H_2 is the binary entropy. The payloads are given in Table 6.1.

6.3.1 LDEAs from the top k insertion test

Figure 6.2.1 shows that for J-MiPOD, a sizable portion of stego images can be detected as stego with only a few inserted 8×8 blocks with stego changes. This is rather surprising because such

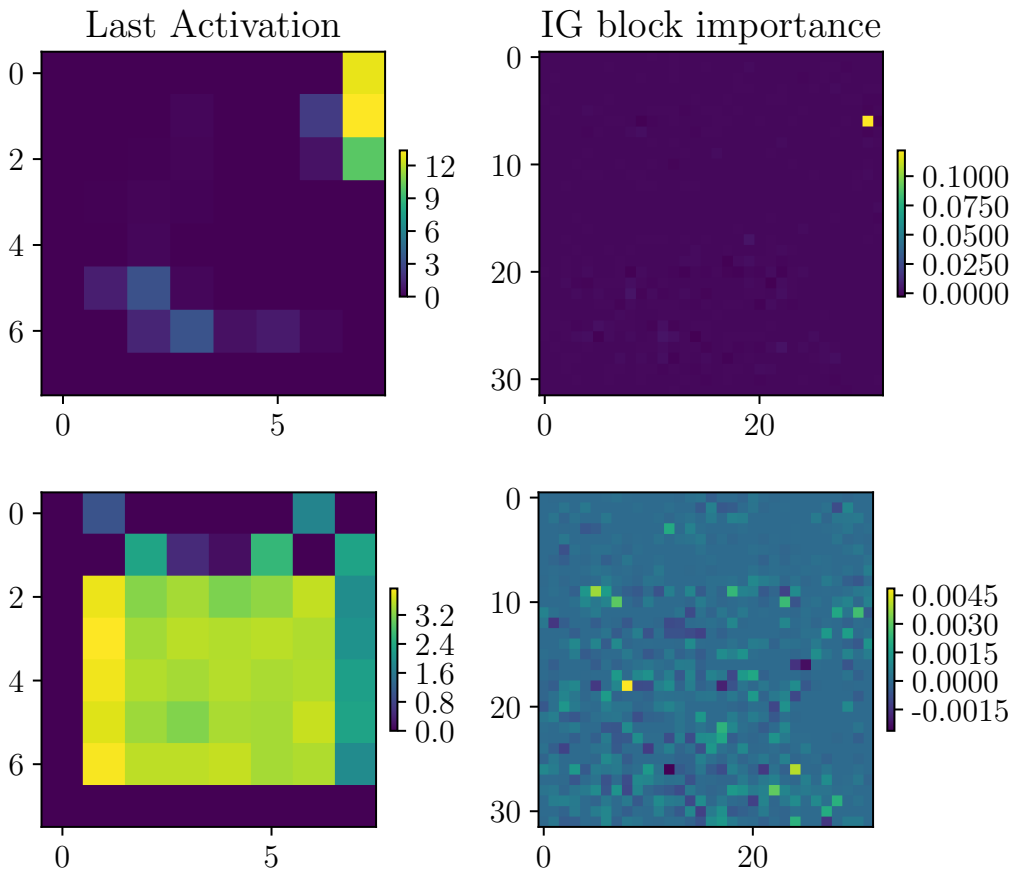


Figure 6.2.2: Last activation (left) and IG block importance map (right) of EfficientNet B4 for image '27938.jpg' embedded with J-MiPOD (top) and J-UNIWARD (bottom). Note that both images are detected as stego with $p_{\text{stego}} = 0.99$ by EfficientNet B4.

	Payload (bpnzac)	P_{E}	MD5	wAUC
J-MiPOD	0.5	.1938	.3837	.9349
J-MiPOD	0.2	.3452	.7033	.8067
J-UNIWARD	0.5	.1967	.4220	.9304
J-UNIWARD	0.2	.3606	.7658	.7792
F5	0.2	.1835	.4292	.9292
−F5	0.05	.0866	.1248	.9827
Jsteg	0.0112	.1315	.2207	.9595

Table 6.1: Detection performance of EfficientNet B4 for stego schemes used in this chapter and a mixture of QFs of 75, 90, and 95.

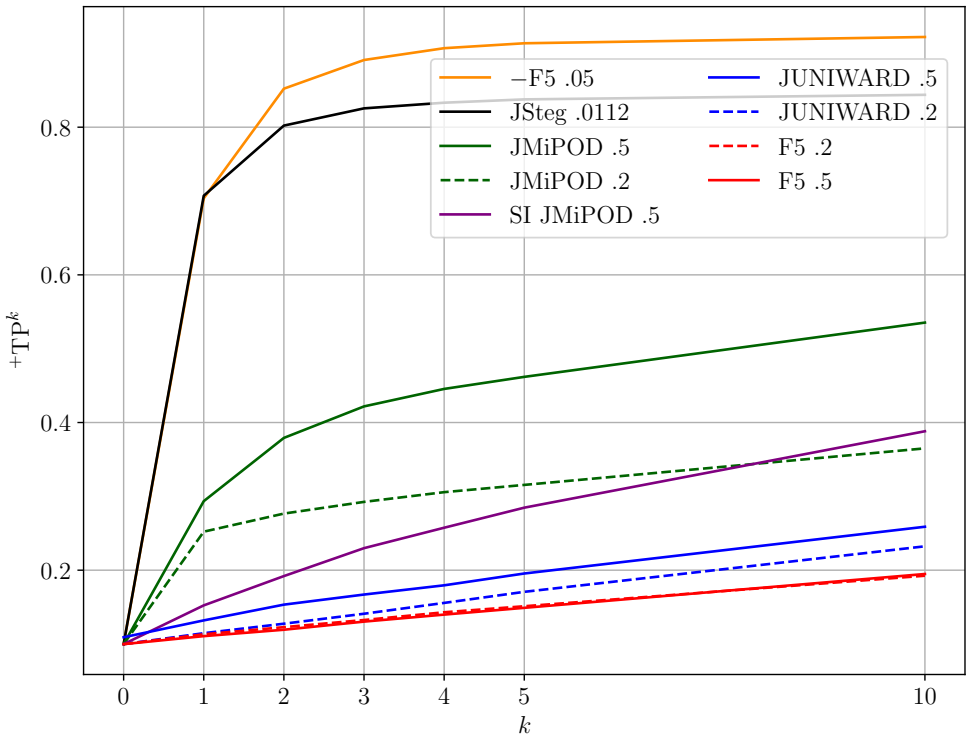


Figure 6.3.1: EfficientNet B4's $^{+}\mathrm{TP}^{k}$ rate as a function of top k inserted stego blocks for various stego schemes.

images have a very small change rate, yet can be detected as stego with high confidence. We say that these images have LDEAs.

Figure 6.3.1 shows the insertion profiles of the same top k insertion test for more embedding schemes with payloads and performance measures shown in Table 6.1. Notice that different embedding schemes have different top k insertion profiles. Also note that the location of such LDEAs in the stego images depends on the actual realization of embedding changes. Different realizations of stego changes for the exact same payload might lead to different LDEAs depending on which DCT coefficients are changed.

The figure also clearly shows that, despite the small payload, Jsteg and –F5 introduce very influential LDEAs as a large percentage of stego images can be identified as stego with only a few blocks with the highest attribution. In contrast, J-UNIWARD and F5 introduce comparatively fewer LDEAs than J-MiPOD. This suggests that for these two algorithms the detector is more an integrator rather than relying on LDEAs.

Conversely, in Figure 6.3.2 we show that reverting the changes in the top k blocks and keeping the rest of the stego image intact (i.e. top k canceling instead of insertion) turns the predicted stegos

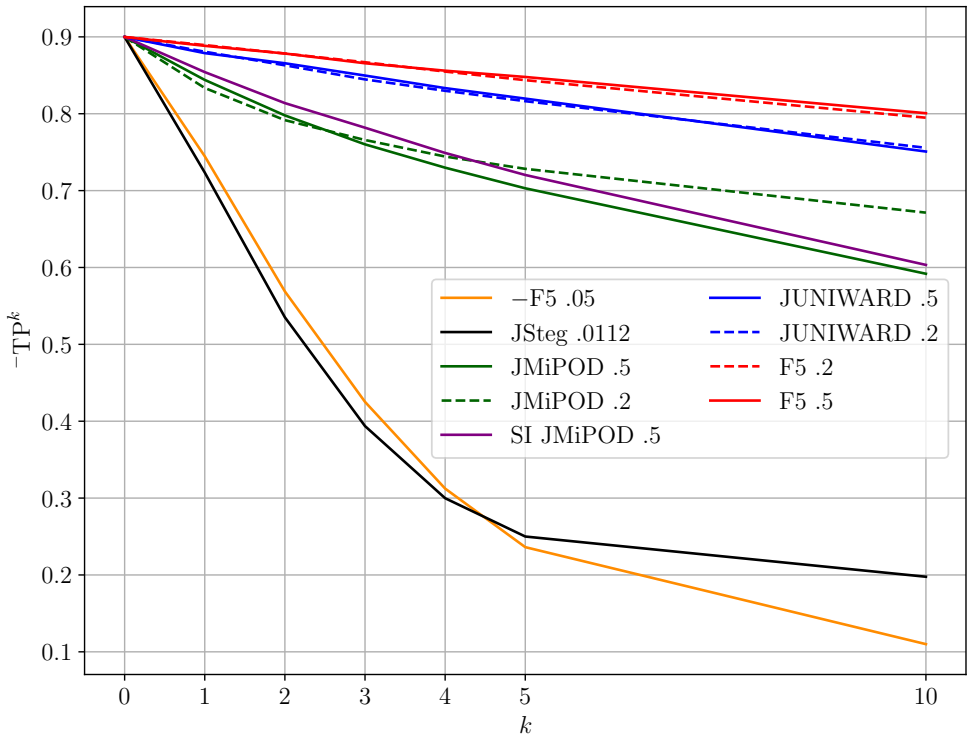


Figure 6.3.2: EfficientNet B4's $^{+}\text{TP}^k$ rate as a function of top k deleted stego blocks for various stego schemes.

into missed detection. Note that the trends are complementary to those observed for the top k insertion test. The decision threshold was set for 90% True Positive rate.

6.3.2 Do Rich Models catch LDEAs?

Next, we contrast CNNs and rich models to find out whether rich models can detect LDEAs with any level of confidence. To this end, we define the concept of a "strong LDEA" to eliminate cases when the cover image already had a score close to the decision threshold. We consider $k = 1$ and adjust the threshold for $^{+}\mathbf{c}^1$ to have a 1% FA rate while keeping the thresholds for FAs at 10% as before. For example, for J-MiPOD at 0.5 bpnzac, images with a strong LDEA must have $f(^{+}\mathbf{c}^1) \geq 0.89$ and $f(\mathbf{c}) \leq 0.55$.

Images with strong LDEAs are usually located at the left-most side of the ROC curve for EfficientNet as shown in Figure 6.3.3, which shows the images with strong LDEAs as red dots. While they are easy to detect by a CNN even when the only changes made in the image are in one block, in contrast, for DCTR+FLD ensemble, LDEAs are not particularly easy to detect as stego images as shown in Figure 6.3.3, where the red dots are scattered rather randomly on the ROC curve. Rich models

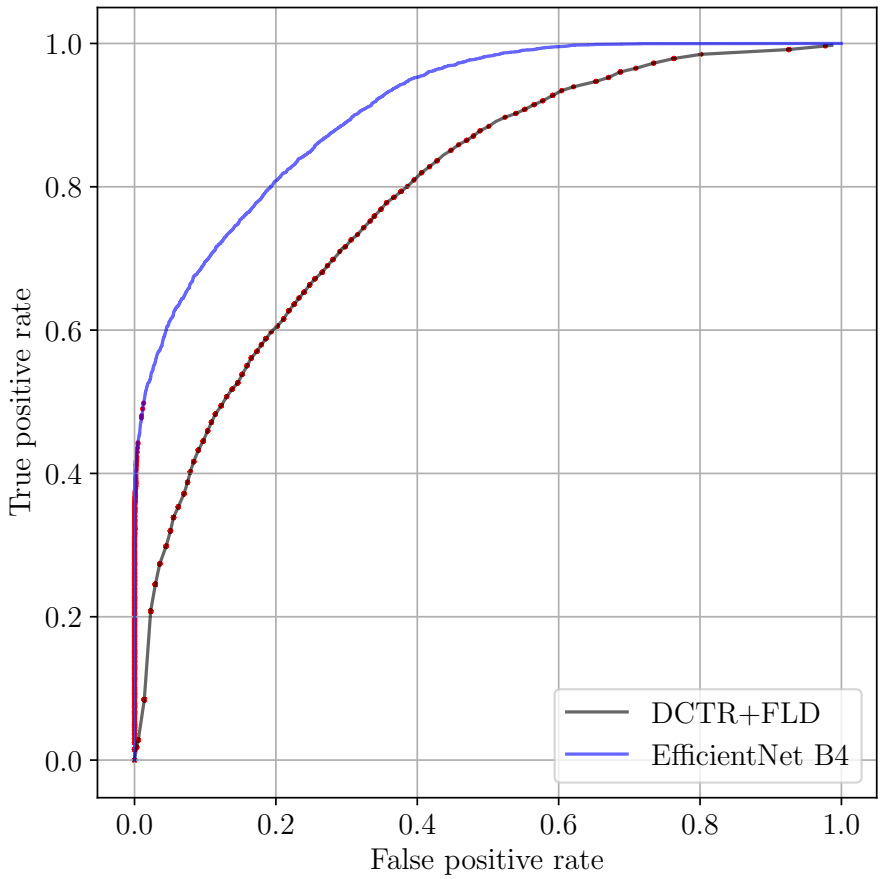


Figure 6.3.3: ROC curve of DCTR+FLD ensemble and EfficientNet B4 for J-MiPOD 0.5 bpnzac. Images with strong LDEAs found using IG and EfficientNet B4 are represented by red dots.

(DCTR in this case) do not catch LDEAs because of their inability to utilize localized artifacts.

6.3.3 Case study 1: J-MiPOD

Figure 6.3.2 and the previous sections discussed the existence of LDEAs introduced by J-MiPOD, which provide overwhelming evidence to a CNN detector to predict the stego class. Figure 6.3.4 shows some examples of LDEAs that are visually identifiable. To further understand the nature of the LDEAs, in Figure 6.3.5 we show the average changes of DCT coefficients in each mode computed over test images containing strong LDEAs. It shows that LDEA blocks have (i) a larger change rate than the average 8×8 block of J-MiPOD (ii) more changes in high frequency DCT coefficients. These coefficients are usually zeros in covers, and changing them to ± 1 creates unnatural artifacts. Figure 6.3.6 shows that, indeed, the LDEA blocks of J-MiPOD have many more changes in zero coefficients than on average. The distribution for J-UNIWARD is given for reference.

Figure 6.3.4: Example of visible local traces of J-MiPOD. The center 8×8 block is the top 1 influential block using IG. Left to right images: '05626.jpg', '47211.jpg', '48020.jpg', and '55961.jpg'.

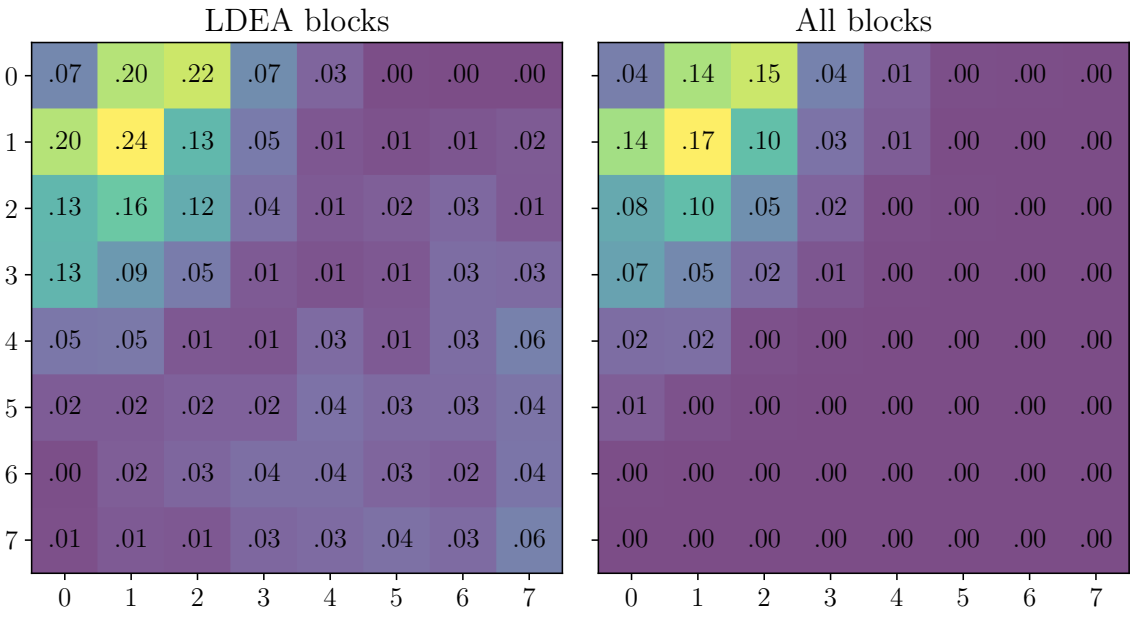


Figure 6.3.5: Average changes per mode for LDEA blocks (left) and over all blocks (right) computed over test images containing strong LDEAs for J-MiPOD.

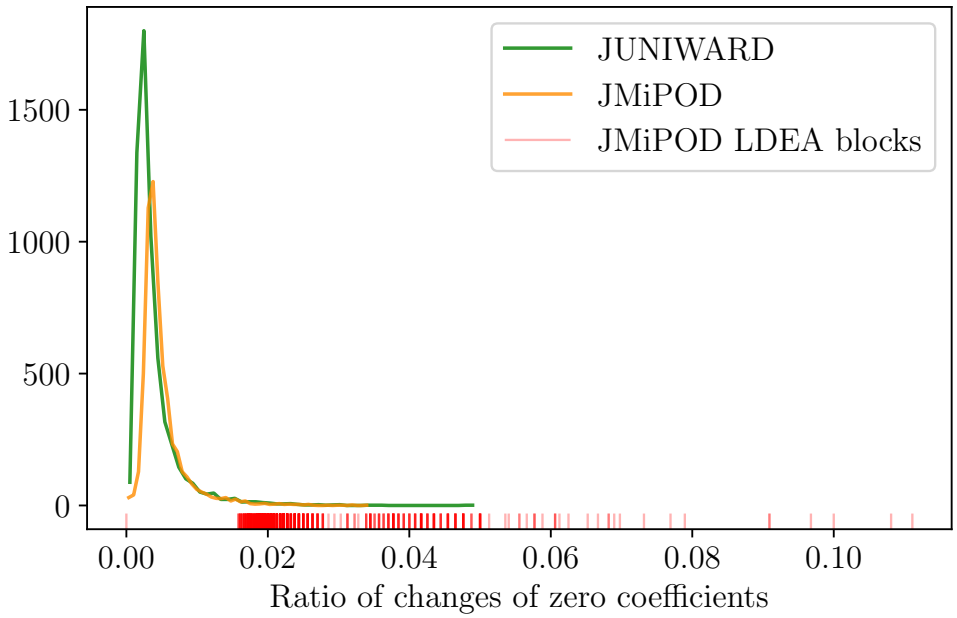


Figure 6.3.6: Histogram of the ratio of changes on zero coefficients w.r.t. all changes for J-UNIWARD and J-MiPOD. Rug plot data points correspond to the same ratio computed only on strong LDEA 8×8 blocks of J-MiPOD.

Additionally, LDEAs transfer between different architectures. For J-MiPOD 0.5 bpnzac 82% of SRNet's images with LDEAs are shared with EfficientNet. Reverting the changes in top 3 influential blocks leads to a substantial increase in Missed Detection (from .3883 to .4975 in terms of MD5 as seen in Figure 6.3.2) for EfficientNet B4, while the DCTR+FLD ensemble missed detection stays mainly unaffected (from .6963 to .6831 in terms of MD5). Moreover, retraining SRNet on a new dataset where the top 3 influential blocks (computed using B4) have been reverted in all images does not bridge that gap and still produces a significantly worse detector (from .4097 to .4878 in terms of MD5).

6.3.4 Case study 2: J-UNIWARD

Figure 6.3.2 shows that J-UNIWARD introduces significantly fewer LDEAs than J-MiPOD even though their detectability is very similar (Table 6.1). In fact, Figure 6.2.2 already shows an interesting difference between the two embedding schemes when looking at their last activation map: J-UNIWARD images tend to activate the majority of the map, whereas J-MiPOD images activate a highly localized area. The ranges of IG block attributions also differ with J-UNIWARD exhibiting a rather spatially uniform attribution map unlike J-MiPOD. This seems to indicate that the network is an "integrator" for most J-UNIWARD images, while it also utilizes localized information for

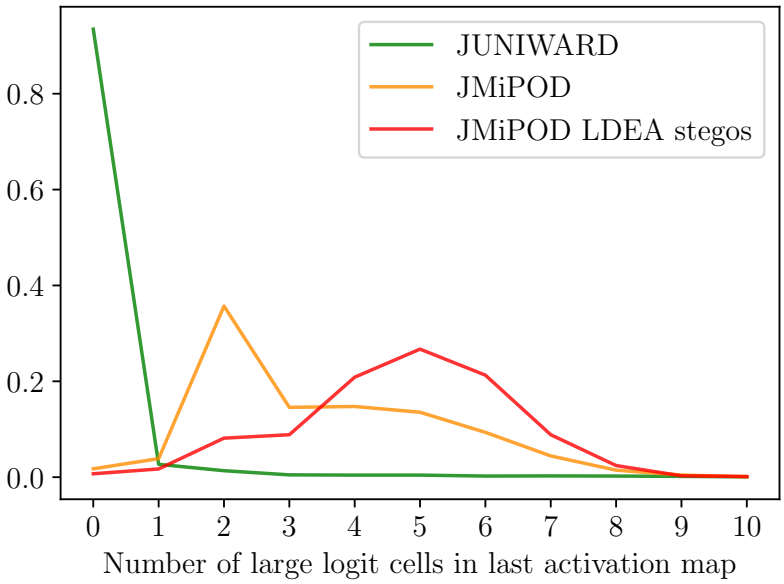


Figure 6.3.7: Normalized histogram of the number of large logit cells of the last activation map of EfficientNet B4 for J-UNIWARD, J-MiPOD, and J-MiPOD with strong LDEAs.

J-MiPOD.

To confirm this conjecture, for each image we count the number of elements in the last activation map that exceed a threshold set as 3× the average of the last activation map (as we try to identify "large logit cells" or spikes in the map). Figure 6.3.7 shows the histograms of these counts across 6,000 test images. For J-UNIWARD, these large logit cells are almost non-existent, while for J-MiPOD and especially J-MiPOD images with strong LDEA blocks many last activation maps are comprised of such spikes.

6.3.5 Case study 3: Jsteg

Figure 6.3.2 shows that Jsteg introduces many LDEAs, which is not surprising since Jsteg is not content adaptive and highly likely to produce detectable artifacts. On average, the top 1 influential blocks of Jsteg have 98.01% of changes *increasing* the absolute value of the DCT coefficients, whereas on average across all blocks Jsteg increases the absolute value of DCT coefficients with a rate of only 65.06%. Increasing the absolute value increases the block variance, which makes it easier to detect in a smooth area.

Figure 6.4.1: Attribution maps for image '43201.jpg' from the ALASKA II dataset embedded with J-UNIWARD and bUERD from the multi-class, and binary J-UNIWARD EfficientNet B4 (from left to right). Notice the anti-correlated attributions in the right boundary of both images from the multi class attributions, which are not visible in the binary attributions. The figure shows only the 90% largest attributions for each map in absolute value for visual clarity.

6.3.6 Case study 4: F5, –F5

Figure 6.3.2 shows that F5 introduces very few LDEAs. Unlike other schemes, F5 only decreases the absolute value of DCT coefficients. For −F5, the LDEAs count is the largest. The culprit is the embedding operation of increasing the absolute value of DCT coefficients as it adds artificial content to 8×8 DCT blocks. This is further confirmed by comparing –F5 with Jsteg with payload scaled to have the same number of changes as −F5. While Jsteg's curve is lower than −F5 for $k > 1$, both have the same number of strong LDEAs (for $k = 1$).

6.4 Multi-class detectors and stego inhibition

In this section, we briefly study a multi-class CNN detector that uses inhibition to distinguish between embedding algorithms. To this end, we purposely selected two embedding algorithms with very different selection channels: J-UNIWARD and bUERD, which is a version of UERD that was used in the ALASKA II competition. An implementation mistake made bUERD's selection channel anomalous with the embedding changes concentrated around the image boundary in most stego images. A network able to see LDEAs should discover this flaw and exploit it for detection.

Our multi-class detector was the EfficientNet B4 trained using multi-class cross-entropy loss. In this section, we drop the modulation by $\mathbf{y} - \mathbf{c}$ in Eq. 6.2.1 since we are interested in blocks with changes by both bUERD and J-UNIWARD not only a by one of them at a time.

Given a J-UNIWARD image, the stego attribution is typically high in blocks with complex content, while an inhibitory attribution at the image boundary. For the bUERD version of the same image,

the attributions at the image boundary are often anti-correlated with the attributions from the J-UNIWARD image. We call this phenomenon "stego inhibition" as the CNN uses artifact traces from all stego schemes it was trained on. Another intuitive explanation of stego inhibition is when the CNN predicts "this is a J-UNIWARD image and not bUERD" using inhibition of bUERD artifacts. An example of this is shown in Figure 6.4.1. We compare this to a known phenomenon in computer vision and neurology where neurons explicitly inhibit against features that do not make sense in certain spatial areas. In this case, the stego noise at the image boundary is representative of bUERD, and does not make sense for images other than bUERD (provided the boundary does not contain complex content). Also notice in Figure 6.4.1 that the attribution of the J-UNIWARD image from both the binary and multi-class detectors have strong similarities (outside the right boundary), which means that both detectors have converged to detecting similar patterns.

6.5 Conclusions

Using attribution tools, we provide evidence for the popular belief that CNNs reach their decision by detecting local embedding artifacts. By analyzing modern content-adaptive schemes and older embedding paradigms, we characterize these artifacts and show that they are mostly associated with high frequency content-creating changes. CNNs ability to leverage localized signals plays a role in distinguishing between selection channels of different embedding algorithms when training a multi-class detector.

Chapter 7

Detector-Informed Batch Steganography and Pooled Steganalysis

The main bulk of work in this field of steganography concerns digital images and focuses on designing embedding algorithms and detectors that perform the best in a single image for a fixed relative payload. In practice, however, the sender can adopt a smarter strategy and distribute the communicated message across multiple covers to decrease the chances of being detected. On the other hand, the Warden is also free to combine evidence from multiple images to decide whether steganography is taking place.

Batch steganography and pooled steganalysis have been originally introduced in [200] together with the so-called shift hypothesis, which claims that the embedding rigidly shifts detector outputs by an amount that depends on the embedded payload size. The first batch strategies [29; 30; 31], which focused on non-adaptive hiding schemes and quantitative detectors, concluded that the payload should either be concentrated in as few covers as possible or spread evenly.

In [30], the author studied pooled steganalysis under the assumption that the Warden knows the chunk sizes but not their assignment to individual images. In a different setup [201], a local outlier factor was used to identify the steganographer from among a large set of users. The topic of learning optimal pooling functions appeared in [202]. Batch steganography with modern content-

adaptive embedding algorithms and three ad hoc batch algorithms was studied in [32]. Adversarial embedding [203] was extended to batches of cover images in [34] but performed poorly against an adversarial-aware Warden. In [204], the authors considered batch steganography in JPEG images of different qualities. The optimal size of the bag for Gaussian batch embedding was studied in [33] without considering pooled steganalysis.

The next section explains the reasoning for the setup of batch steganography and pooled steganalysis studied in this chapter. To further motivate our work, in Section 7.2 we demonstrate that the often adopted shift hypothesis is no longer valid for content-adaptive embedding, a fact that holds for the previous generation of detectors built around rich models and linear classifiers as well as modern detectors built as Convolutional Neural Networks (CNNs). In the same section, we show that detectors exhibit highly non-Gaussian distribution on covers. Section 7.3 contains a formal mathematical description of three pooled detectors considered in this chapter. Two novel detector-informed batch steganographic techniques are described and theoretically analyzed in Section 7.4. The setup of our experiments, including implementation details, is explained in Section 7.5. The results of all experiments together with their interpretation and discussion appear in Section 7.6. The chapter is concluded in Section 7.7.

7.1 Basic setup

In batch steganography, two actors, Alice and Bob, exchange messages hidden in digital images. To avoid being detected by an adversary (the Warden), they use modern content-adaptive spatial-domain steganography and adjust the payload size embedded in each image to decrease the risk of being detected. The Warden combines the outputs of a single-image detector applied to all images exchanged by Alice and Bob in an effort to discover the use of a steganographic channel and not necessarily identify which images are cover and stego.

This problem of batch steganography and pooled steganalysis may accept many different forms depending on what information about the cover source, the steganographic method, the payload spreading strategy, and possibly Warden's detector is available to all actors. Following Kerckhoffs's principle, we are mainly interested in the situation when the Warden has full knowledge of algorithms used by Alice and Bob but not any shared secret or specific data used by the senders. In particular, we assume that the steganographers and the Warden have access to the same source of covers, which they can use in any way to design a payload spreading strategy as well as build detectors.

We will also assume that the Warden knows the steganographic method that might be in use and the payload-spreading strategy. For example, if the steganographers use feedback from a detector to determine the size of payload chunks embedded in each image, the Warden can train the same detector architecture on her end but it will ultimately be a slightly different detector because of different training data. Moreover, the payload chunk sizes will also generally depend on the cover images to which the Warden does not have access.

Having said this, we will at times consider a payload-aware Warden that has access to the exact payload chunk sizes that Alice sends as a form of a worst case scenario and to evaluate the impact of the lack of such precise knowledge.

While the steganographers may opt for a spreading strategy that is free of any assumptions about Warden's detector, such as the Image Merging Sender (IMS) and Detectability / Distortion Limited Senders (DeLS / DiLS) considered in [32], they are free to guess and make use of a detector that is likely to be used by the Warden or any other detector. The specific assumptions made in this chapter will be clarified later based on discussions and other important experimental facts concerning content-adaptive embedding and modern steganalysis detectors.

7.2 New context

The problem of batch steganography and pooled steganalysis has been revisited many times throughout the history of this field. In this section, we challenge some of the assumptions made in prior art to motivate our approach.

In [32], an argument based on the Central Limit Theorem (CLT) was made that, on cover images, the outputs of a single-image detector that uses high-dimensional rich models and a linear classifier is zero-mean Gaussian. Leveraging the shift hypothesis, the authors further assumed that the embedding merely shifts this distribution by an amount that depends on the embedded payload. The Gaussianity and the shift hypothesis allowed the authors to derive an optimal pooled detector in the form of a matched filter, which in practice correlates detector outputs with shifts estimated from near embedding invariants and the payload itself. Within this context, they studied the IMS and DeLS (DiLS), the last two spreading so that the same level of detectability (distortion) is induced in every image.

Below, we demonstrate that modern detectors not only exhibit highly non-Gaussian behavior but also clearly fail to satisfy the shift hypothesis. This is true for both non-adaptive and content-

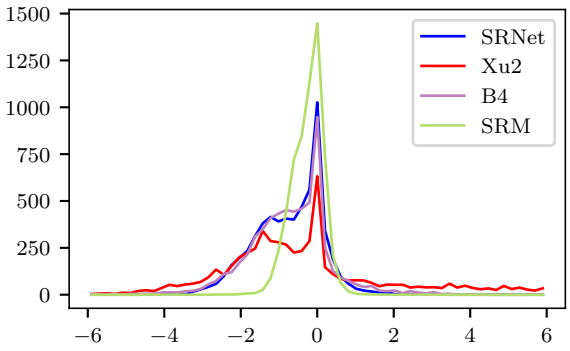


Figure 7.2.1: Distribution of the soft detector output for SRNet, EfN B4, Xu2, and SRM with LCLC trained on covers vs. uniform payload mixture of HILL.

adaptive steganography and detectors based on rich models as well as CNNs. For better readability, the description of datasets and detectors, including their training for all experiments commented upon in this section is postponed to Section 7.5.

7.2.1 Non-Gaussian distribution on covers

Figure 7.2.1 shows the distribution of soft outputs of four detectors on 256×256 grayscale cover images from ALASKA II when training them as binary detectors on cover versus stego images embedded with a uniform mixture of payloads from $\{0.05, 0.1, 0.2, \ldots, 1.4, 1.5\}$ bpp. The soft output for the Spatial Rich Model (SRM) [65] implemented with the Low Complexity Linear Classifier (LCLC) [73] is the projection of the rich feature on the weight vector. For the three CNNs, SRNet [99], Efficient Net B4, and SE-ResNet18 (Xu2 net), the output is the logit. The cover distribution for all detectors is highly asymmetric and spiky. The distribution on covers is also clearly non-Gaussian and bimodal for the networks with the left "hump" corresponding to "easy covers."

While the fact that CNNs produce highly non-Gaussian outputs on both cover and stego images is less surprising due to their inherent non-linearity, rich model features are also non-linear functions of the image since they are higher-order statistics (histograms) of quantized and truncated noise residuals.

7.2.2 Failure of the shift hypothesis

Figure 7.2.2 shows the distribution of two of the above four detectors on stego images embedded with a range of fixed relative payloads. With increased payload size, the distribution gradually "morphs"

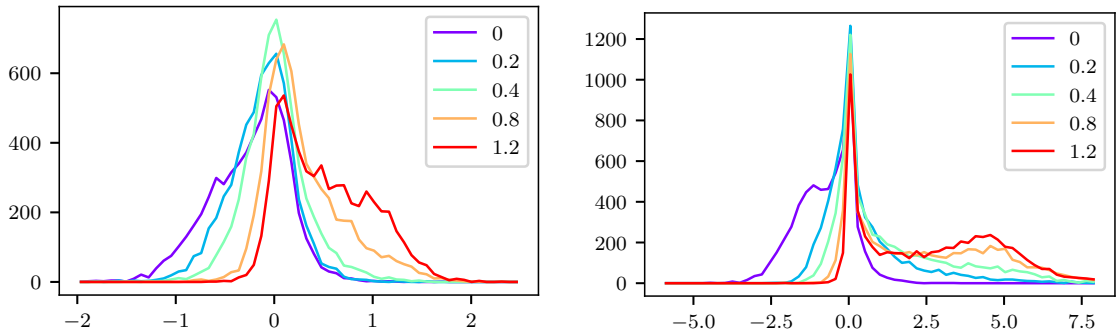


Figure 7.2.2: Distribution of detectors' soft output on cover and stego images embedded with HILL for a fixed relative payload. Left: LCLC with SRM, right: SRNet. Note that the distributions morph in a far more complex manner than a simple shift.

to the right, affecting mostly the distribution tails, while the peak at zero stays nearly stationary. In fact, in order to obtain a rigidly shifted distribution, one would need to adopt a non-trivial spreading strategy (see Section 7.4). The shift hypothesis, as originally conceived in [200], is likely limited to quantitative detectors since their expected test statistic is the change rate (payload).

7.2.3 Complex response curves

Undoubtedly, the key element in considering the problem of spreading payload across images is the response of the Warden's detector as a function of the payload size – the detector's response curve – which depends on the cover image and the steganographic method. A cover image with a completely flat response curve would be ideal for embedding a large payload as the embedding is "invisible" to the detector. And this is true regardless of whether it is detected as cover or stego. On the other hand, an image exhibiting a steep response curve should hold a comparatively smaller payload.

Since embedding a secret message is a stochastic process, the detector response naturally exhibits a statistical spread, which increases with increased payload (see Figure 7.2.3). To eliminate this source of randomness, we define the response curve (RC) for a given cover image and detector as the expected value of the response over embeddings with different stego keys (seeds for an embedding simulator). In Figure 7.2.3, the RCs are rendered with thick blue lines obtained by averaging over 100 embeddings for each payload with the light blue shade used to depict the standard deviation. The diversity of these responses is responsible for the failure of the shift hypothesis.

Note that RCs are mostly non-decreasing with the exception of a few images for which the response decreases for very large payloads (e.g., image 10518). Despite the slight drop, the final class label

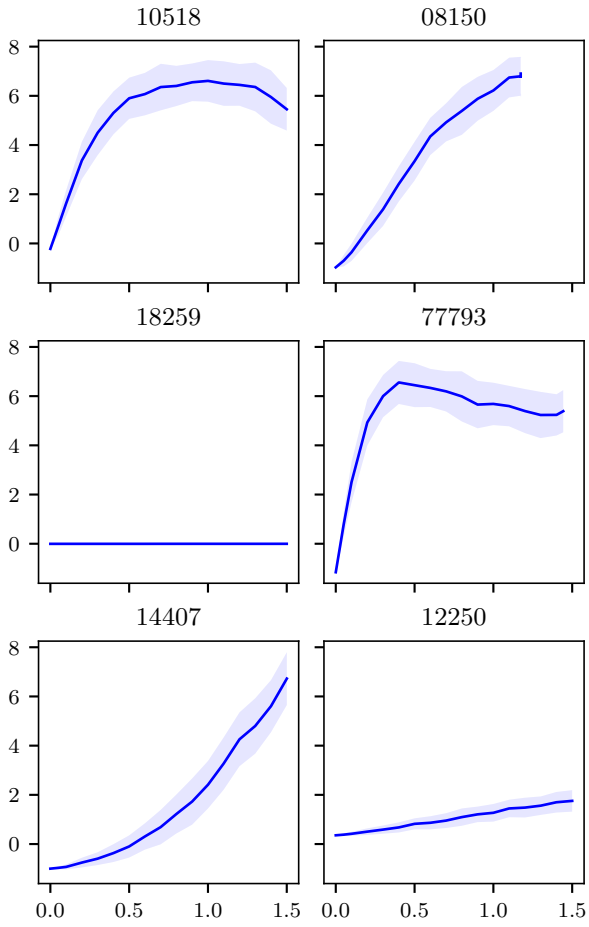


Figure 7.2.3: SRNet's response curves (the logit as a function of embedded payload size) for six selected images. The solid line is the expectation over 100 embeddings with different stego keys with the light blue shade used to depict the standard deviation. Detector: SRNet trained on uniform payload mixture for HILL.

is unlikely to flip because the logit values are still very large. While we are not certain why this is happening, it might be due to the fact that the content-dependent stego noise for large payloads might start resembling sensor noise in some images. To simplify our reasoning, we adopt the feasible assumption that all RCs are non-decreasing for the entire payload range.

The RCs tell the tale of what is happening at detection. For image 14407, the RC is initially flat and then steeply bends upwards. This is likely because the image contains some complex content where the detection is difficult, and once the embedding "spills over" to other parts of the image due to increased payload, it quickly starts contributing to detectability. The flat curves of images 18259 and 12250 mean that they can hold a very large payload without changing the detector's output. Lastly, we point out the steep response curves for images 10518, 77793, and 08150 with smooth content where embedding quickly becomes very detectable. Note that for image 08150, the maximal embeddable payload is only about 1.2 bpp because the image contains wet pixels [205].

Figure 7.2.4 shows the RCs for the same six images for four different detectors. Note that while the network detectors are very different deep architectures, the response curves exhibit qualitative similarities. This justifies our choice to use detector output as feedback for batch steganography.

Finally, we remark that the steganographers must select their images randomly from their cover source as any cover selection or rejection would skew the statistics of the cover source, a fact that would be detected by the Warden who is testing whether her detector's outputs are consistent with the detector distribution on covers. Thus, the best the sender can do is to minimize the disturbance to the distribution of Warden's test statistic. We come back to this problem in the next section when we lay out a more detailed formulation of our setup.

7.3 Pooled steganalysis

In this section, we describe three pooling strategies for the Warden that will be used to assess security of batch steganography in this chapter.

We will assume that the steganographers maintain an average payload per pixel $r \in [0, \log_2 3]$, the communication rate. By the square root law [206], this means that, asymptotically, they will be caught with near certainty. Our goal is not perfect or bounded security, which would require the communication rate to taper off to zero, but to minimize the detectability in each bag of images. For simplicity, in the rest of this chapter we assume that the Warden knows r and that the embedding method is fixed and known to all actors.

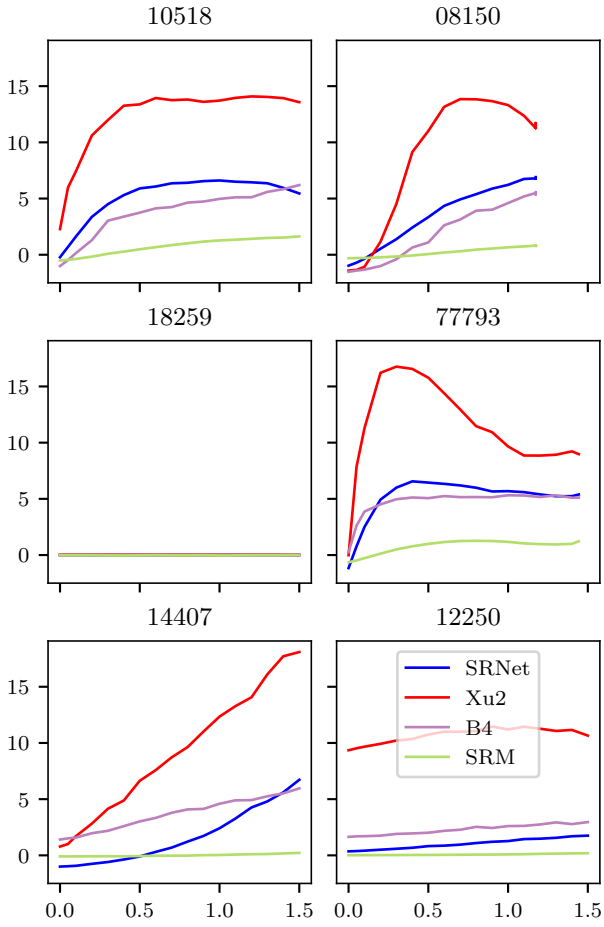


Figure 7.2.4: Response curves for the same six selected images as in Figure 7.2.3 and four different detectors.

Let $\mathcal{C}$ denote the set of all possible cover images of some fixed size. A cover bag of size B, $\mathbf{C} = (\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(B)})$, is formed by selecting B cover images $\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(B)} \in \mathcal{C}$ according to some probability distribution over $\mathcal{C}$. A spreading strategy S induces a unique mapping $\alpha_{r,S} : \mathcal{C}^B \to [0, \log_2 3]^B$ that determines the relative payloads (in bpp) embedded in the B images using a ternary steganographic scheme. When r and S are clear from context, we simply write $\alpha_i \in [0, \log_2 3]$ to denote the ith component of $\alpha_{r,S}(\mathbf{C})$, i.e., the relative payload embedded in the ith image. The map $\alpha_{r,S}$ must satisfy the payload constraint

$$\sum_{i=1}^{B} \alpha_i = rB. \tag{7.3.1}$$

A payload tag for rate r is the relative payload $\tau_{r,S}(\mathbf{x}_0^{(i)})$ that the ith image receives for an infinitely large bag.

A single-image detector is a mapping $d : \mathcal{C} \to \mathbb{R}$ that assigns to each image a soft output from the detector. The soft outputs can be thresholded for a hard cover / stego decision based on application-dependent requirements, such as controlling the false alarm rate. The response curve for image $\mathbf{x}_0^{(i)}$ and detector d is the function $\varrho_i : [0, \log_2 3] \to \mathbb{R}$

$$\varrho_i(\alpha) = \mathbb{E}[d(\mathbf{x}_\alpha^{(i)})] \tag{7.3.2}$$

obtained as the expected value of d on stego images $\mathbf{x}_\alpha^{(i)}$ when embedding cover $\mathbf{x}_0^{(i)}$ with payload α with random messages and stego keys. To distinguish the mathematical objects used by the Warden from those used by the steganographers, we will use the superscript 'W' for the Warden and 'S' for the steganographers. Pooled detectors will be denoted with the Greek letter π.

Let f_0^{W} denote the Warden's detector distribution on covers (c.f., Figure 7.2.1). Given B images $\mathbf{y}^{(i)}$, $i = 1, \ldots, B$, communicated by the sender and under inspection by the Warden, $\mathbf{Y} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(B)})$, the Warden computes $d^{\mathrm{W}}(\mathbf{y}^{(i)})$ for all images and infers whether the sender uses steganography. In the absence of any other knowledge about the spreading strategy or the communication rate, the Warden would face a composite hypothesis test, namely testing for a known distribution:

$$\begin{aligned} \mathrm{H}_0 : &\quad d^{\mathrm{W}}(\mathbf{y}^{(i)}) \sim f_0^{\mathrm{W}} \quad \text{for all } i \\ \mathrm{H}_1 : &\quad d^{\mathrm{W}}(\mathbf{y}^{(i)}) \nsim f_0^{\mathrm{W}} \quad \text{for some } i. \end{aligned} \tag{7.3.3}$$

Note that in this chapter, unlike the rest of this dissertation, the letter $\mathbf{y}$ does not denote a stego image, but an image under inspection by the Warden, which can potentially be a stego or a cover

image. Stego images embedded with payload α are denoted $\mathbf{x}_\alpha$ and cover images are denoted $\mathbf{x}_0$ as described above.

7.3.1 Correlator pooling

Since we assume that the Warden knows the spreading strategy and the communication rate r, she can test for an increase in the detector response $s_i = \varrho_i^{\mathrm{W}}(\alpha_i) - \varrho_i^{\mathrm{W}}(0)$. However, since she does not have access to cover images, she needs to estimate the response on the cover, $\varrho_i^{\mathrm{W}}(0)$, or simply add it to the modeling error. Moreover, a realistic Warden will also need to estimate the payloads α_i from the images at hand. In particular, she can obtain the estimated payload $\hat{\alpha}_i$ possibly embedded in $\mathbf{y}^{(i)}$ by computing the ith component of $\alpha_{r,S}(\mathbf{Y})$. The statistical hypothesis testing problem thus becomes

$$\begin{aligned} \mathrm{H}_0: &\quad d^{\mathrm{W}}(\mathbf{y}^{(i)}) = \xi_i &&\text{for all } i \\ \mathrm{H}_1: &\quad d^{\mathrm{W}}(\mathbf{y}^{(i)}) = \hat{s}_i + \xi_i &&\text{for all } i, \end{aligned} \tag{7.3.4}$$

where $\hat{s}_i = \hat{\varrho}_i^{\mathrm{W}}(\hat{\alpha}_i) - \hat{\varrho}_i^{\mathrm{W}}(0)$ is the estimated expected increase of the detector output using a RC $\hat{\varrho}_i^{\mathrm{W}}(\alpha)$ computed from the image at hand $\mathbf{y}^{(i)}$, $\hat{\varrho}_i^{\mathrm{W}}(\alpha) = \mathbb{E}[d^{\mathrm{W}}(\mathbf{y}_\alpha^{(i)})]$, where $\mathbf{y}_\alpha^{(i)}$ is image $\mathbf{y}^{(i)}$ embedded with relative message α, and ξ_i is the modeling error.

In the simplest case of independent Gaussian noise samples $\xi_i \sim \mathcal{N}(0, \sigma_i^2)$, the optimal detector would be the generalized matched filter (correlator) [207]. In our work, we experimented with several different forms of the estimators, including the estimator used in [32] that used near embedding invariants. In the end, the best overall performance was achieved with a pooled detector in the form of a correlator

$$\pi_{\mathrm{COR}}(\mathbf{Y}) = \sum_{i=1}^{B} d^{\mathrm{W}}(\mathbf{y}^{(i)})\gamma(\hat{\alpha}_i), \tag{7.3.5}$$

where $d^{\mathrm{W}}(\mathbf{y}^{(i)})$ are detector outputs on the analyzed images and $\gamma(\alpha)$ is a logistic fit to embedding shifts $\hat{\varrho}_i^{\mathrm{W}}(\alpha) - \hat{\varrho}_i^{\mathrm{W}}(0)$ across a dataset of cover images i.

7.3.2 LRT pooling

Another possibility for the Warden is to test whether the detector output for the ith image is consistent with the distribution of her detector $f_{\hat{\alpha}_i}^{\mathrm{W}}$ on stego images all embedded with the same

relative payload $\hat{\alpha}_i$:

$$\begin{aligned} \mathrm{H}_0: &\quad d^{\mathrm{W}}(\mathbf{y}^{(i)}) \sim f_0^{\mathrm{W}} \quad \text{for all } i \\ \mathrm{H}_1: &\quad d^{\mathrm{W}}(\mathbf{y}^{(i)}) \sim f_{\hat{\alpha}_i}^{\mathrm{W}} \quad \text{for all } i \end{aligned} \tag{7.3.6}$$

with the optimal pooled detector being the log-likelihood ratio

$$\pi_{\mathrm{LRT}}(\mathbf{Y}) = \sum_{i=1}^{B} \log \frac{f_{\hat{\alpha}_i}^{\mathrm{W}}\left(d^{(w)}(\mathbf{y}^{(i)})\right)}{f_0^{\mathrm{W}}\left(d^{(w)}(\mathbf{y}^{(i)})\right)}. \tag{7.3.7}$$

7.3.3 Tag based pooling

We also consider the pooling strategy where the Warden makes use of the tags $\tau_i = \tau_{r,S}(\mathbf{x}_0^{(i)})$ and trains her single-image detector as a binary classifier between covers and stego images embedded with tags. For large enough bags, this is the correct stego source from which the sender draws their images. Note that the stego source only depends on the spreading strategy S and rate r. We make an argument that, for large bags and for images in the bags selected randomly, the optimal pooling strategy is averaging the detector logits. This is because all three deep learning architectures used in this chapter apply average pooling[1] in the last convolutional layer before the fully connected layer. If averaging the detector outputs across all images in the bag was not the best strategy, one could obtain a better single-image detector by splitting each image into smaller tiles, applying the network to the tiles and learning a more sophisticated strategy for combining the outputs. In summary, for a tag-based single-image detector t^{W}, our pooling strategy is

$$\pi_{\mathrm{TAG}}(\mathbf{Y}) = \frac{1}{B}\sum_{i=1}^{B} t^{\mathrm{W}}(\mathbf{y}^{(i)}). \tag{7.3.8}$$

7.3.4 Average pooling

As the last option considered in this chapter, we added a fourth baseline pooling strategy in the form of a simple average of detector outputs on analyzed images $\mathbf{y}^{(i)}$:

$$\pi_{\mathrm{AVG}}(\mathbf{Y}) = \frac{1}{B}\sum_{i=1}^{B} d^{\mathrm{W}}(\mathbf{y}^{(i)}). \tag{7.3.9}$$

We also experimented with the max pooling strategy $\pi_{\mathrm{MAX}}(\mathbf{Y}) = \max_i d^{\mathrm{W}}(\mathbf{y}^{(i)})$ but do not report on it because it performed very poorly w.r.t. the other strategies.

[1] The word 'pooling' not to be confused with pooling as used in pooled steganalysis.

Note that in this chapter, we will consider both a payload-aware Warden that knows the senders' payloads α_i as well as a realistic Warden that needs to estimate both from the image at hand.[2]

7.4 Batch steganography

In this section, we describe two types of detector-informed spreading strategies depending on the adopted statistical model for the cover source. We also provide theoretical analysis of both senders under certain simplifying assumptions. This analysis will be used to better understand and interpret the experimental results in Section 7.6. The sender's single-image detector will be denoted d^{S}.

7.4.1 Shift limited sender

The Shift Limited Sender (SLS) enforces the shift hypothesis by considering the impact of the embedding on the statistical distribution of detector outputs across cover images. To embed an average communication rate r in B cover images $\mathbf{x}_0^{(i)}$, the SLS sender finds the smallest $\delta > 0$ that leads to the same expected detector output shift when embedding payload α_i in $\mathbf{x}_0^{(i)}$, satisfying $\sum_{i=1}^{B} \alpha_i = rB$, and

$$\delta = \varrho_i^{\mathrm{S}}(\alpha_i) - \varrho_i^{\mathrm{S}}(0) \tag{7.4.1}$$

for all i for which $\varrho_i^{\mathrm{S}}\left(\alpha_{\max}(\mathbf{x}_0^{(i)})\right) - \varrho_i^{\mathrm{S}}(0) \geq \delta$, where $\alpha_{\max}(\mathbf{x}_0^{(i)}) \leq \log_2 3$ is the maximal embeddable payload in $\mathbf{x}_0^{(i)}$ equal to the relative number of non-wet pixels. For images that do not satisfy this condition (images with flat response curves), we set $\alpha_i = \alpha_{\max}(\mathbf{x}_0^{(i)})$.

As explained in Section 7.5 in more detail, the SLS was implemented numerically using unidirectional search for δ with the image response curves.

To obtain better insight, below we derive a closed form for the payload by adopting a linear model for response curves:

$$\varrho_i^{\mathrm{S}}(\alpha_i) - \varrho_i^{\mathrm{S}}(0) = b_i \alpha_i, \tag{7.4.2}$$

with $b_i > 0$. This means that we essentially assume that the RCs are not completely flat, and we ignore the upper bound on $\alpha_i \leq \alpha_{\max}(\mathbf{x}_0^{(i)})$.

Since the SLS requires $b_i \alpha_i = \delta$ for all images in the bag, the payload constraint (7.3.1) implies that

[2] More on this appears in Section 7.6.2.

$\delta = rB/\sum_{i=1}^{B} 1/b_i$, which gives us the following expression for α_i

$$\alpha_i = \frac{rB}{b_i \sum_{k=1} \frac{1}{b_k}}. \tag{7.4.3}$$

7.4.2 Minimum deflection sender

The Minimum Deflection Sender (MDS) considers a statistical model for *each scene* rather than across images. The specific cover used by the sender is a sample from an acquisition oracle taking images of the same scene with the same acquisition device. Sensor noise and possibly small spatial shifts and rotations due to camera shake would contribute to the randomness.

The main advantage of this approach is that the detector output on such cover images is well modeled by a Gaussian distribution due to the fact that the detector can be linearized on the neighborhood of the noise-free cover image. We assume that the embedding changes the expectation of the detector output based on the response curve but does not change the variance. Hence, the sender determines the payloads to minimize the power of the most powerful detector for the following hypothesis testing problem:

$$\begin{aligned} \mathrm{H}_0: &\quad d^{\mathrm{S}}(\mathbf{y}^{(i)}) \sim \mathcal{N}(\mu_i, \sigma_i^2) \quad \text{for all } i \\ \mathrm{H}_1: &\quad d^{\mathrm{S}}(\mathbf{y}^{(i)}) \sim \mathcal{N}(\mu_i + s_i, \sigma_i^2) \quad \text{for all } i, \end{aligned} \tag{7.4.4}$$

where μ_i is the expected value of d^{S} on cover images generated by the acquisition oracle for the ith image and s_i is the expected increase of detector response due to embedding payload α_i. Note that in (7.4.4) we assume that the acquisition variance dominates the variance due to embedding a random message, hence the variances are equal under both hypotheses. For a clairvoyant Warden who uses the same detector $d^{\mathrm{W}} = d^{\mathrm{S}}$ and knows μ_i and σ_i^2, with cover images drawn independently from the cover source, the most powerful detector is the likelihood ratio test, which assumes the form of a mean-shifted Gauss-Gauss problem. Thus, its performance is determined by the deflection coefficient $\sum_{i=1}^{B} s_i^2/\sigma_i^2$.

For practical implementation, we will assume that $d^{\mathrm{S}}(\mathbf{x}_0^{(i)}) = \varrho_i^{\mathrm{S}}(0) \approx \mu_i$ is approximately equal to the expected detector output across all acquisitions. Hence, the MDS selects the α_i to be embedded in $\mathbf{x}_0^{(i)}$ that minimizes the deflection[3]

$$\Delta^2 = \sum_{i=1}^{B} \frac{\left(\varrho_i^{\mathrm{S}}(\alpha_i) - \varrho_i^{\mathrm{S}}(0)\right)^2}{\sigma_i^2}. \tag{7.4.5}$$

[3]As explained in Section 7.5, for the MDS we use a logistic fit to the RCs instead of the RCs to allow for a more efficient gradient descent based search algorithm.

While our assumptions about Warden's access to d^{S} and μ_i and σ_i^2 are too idealistic, we can claim that the MDS considers the worst case scenario. Since estimating the variances σ_i^2 experimentally using the acquisition oracle would be far too elaborate and even infeasible in many cases, we further simplify the MDS by assuming that the variances σ_i^2 are all approximately the same. Experiments on Monobase [134] with simulated acquisition at higher ISO (as in Natural Steganography [134]) confirmed that the detector variance is indeed rather stable across different scenes.

To obtain insight into how the MDS assigns payloads, we again derive a closed form expression for α_i using the linear model (7.4.2) for the response curves $\varrho_i^{\mathrm{S}}(\alpha_i) - \varrho_i^{\mathrm{S}}(0) = b_i\alpha_i$. To minimize the deflection Δ^2 with equal variances $\sigma_i^2 = \sigma^2$ subject to the payload constraint (7.3.1), we find the stationary point of the Lagrangian

$$\mathscr{L} = \frac{1}{2}\sum_{k=1}^{n} b_k^2\alpha_k^2 - \lambda\left(\sum_{k=1}^{n}\alpha_k - rB\right), \tag{7.4.6}$$

which yields the closed form for MDS payloads

$$\alpha_i = \frac{rB}{b_i^2\sum_{k=1}^{B}\frac{1}{b_k^2}}. \tag{7.4.7}$$

7.5 Implementation

In this section, we list the details regarding our implementation of the batch steganography algorithms as well as the detectors.

7.5.1 Datasets

The dataset is the ALASKA II split into three parts (Split 1, 2, and 3), each containing 25,000 images further split into 22k, 1k, and 2k images for training, validation, and testing. The splits are used to study the impact of a mismatched training set for training Warden's detector. The images were developed as in [43] without the final JPEG compression step. Alice uses the test set of Split 1 to send her secret messages in bags of size B by sampling B images with replacement.

Because of the sheer amount of possible combinations of the steganographer's detector, the Warden's detector, stego schemes, communication rates r, bag sizes, and spreading / pooling strategies, we limit our exposition to the steganographic scheme HILL[4] and mainly the rate $r = 0.3$ bpp. Instead

[4]In particular, since we observed qualitatively and quantitatively similar conclusions for MiPOD, the results are

of reporting the complete set of results for all possible setups, we highlight the most interesting and relevant findings.

7.5.2 Single-image detectors

For spreading, the sender uses a single-image detector d^{S} in the form of an SRNet (SRNet1) trained on Split 1. Splits 2 and 3 are used by the Warden who will train d^{W} as another instance of SRNet (SRNet2) on Split 2, Xu2 on Split 2, EfN B4 on Split 3, and SRM on Split 3. EfN B4 and Xu2 were modified by removing the average pooling and strides from the first two layers as described in [2]. All network detectors are pre-trained on ImageNet, SRNet was pre-trained on a binary task of steganalyzing J-UNIWARD [19] (the so-called JIN pre-training exactly as described in [120]), while the other networks were pre-trained on the ImageNet classification task.[5] Steganalysis training on HILL / MiPOD is done with relative payloads randomly drawn from the uniform distribution on the set of relative payloads $\mathcal{P} = \{0.05, 0.1, 0.2, \ldots, 1.4, 1.5\}$.

We also add another, qualitatively different single-image detector based on the Spatial Rich Model (SRM) [65] and the LCLC, also trained on payloads randomly uniformly drawn from $\mathcal{P}$.

7.5.3 Pooled detectors

For the correlator pooling strategy, the Warden uses her test set to fit a logistic curve to the embedding shifts $\varrho_i^{\mathrm{W}}(\alpha) - \varrho_i^{\mathrm{W}}(0)$ to obtain $\gamma(\alpha)$. The logistic curve is defined as

$$p(x) = \frac{a}{1 + e^{c(x-m)}} + h, \tag{7.5.1}$$

with $0 < a$, $m < \infty$, $-\infty < c < 0$, $h \in \mathbb{R}$, and the fit is performed using non-linear least squares[6] initialized at $(a, m, c, h) = (1, 1, -1, 0)$.

For the LRT pooling strategy, the Warden embeds her test set with a set of relative payloads $\mathcal{P} = \{0.05, 0.1, 0.2, \ldots, 1.4, 1.5\}$. Then she proceeds to estimate the distribution of the detector's output f_α^{W} for each $\alpha \in \mathcal{P}$.[7] To cover the entire range of possible payloads, the Warden linearly interpolates between likelihoods evaluated at the payload grid $\mathcal{P}$.

not reported.

[5] Downloaded from https://github.com/rwightman/pytorch-image-models

[6] Using scipy's curve_fit function

[7] Using scipy's gaussian_kde function

For the tag-based poolers, the Warden fine-tunes her single image detectors on a dataset embedded with tags computed by randomly grouping all training images into bags of $B = 100$. Note that the Warden has to train a tag-based pooler for each spreading strategy and average communication rate.

7.5.4 Senders

The IMS was implemented by considering a given bag of B images each with N pixels as a single large image into which the total payload of rBN bits was embedded using an embedding simulator. The costs were pre-computed from single images. We would like to point out that this version of the IMS differs from the implementation used in [32]. There, the authors first pre-computed tags for all images from their dataset and then simply selected B images for a given bag. Thus, the communication rate r varied from bag to bag, and was maintained across bags only in expectation. This difference is rather important as will become apparent when studying the detectability as a function of B.

The SLS was implemented by searching for the smallest δ satisfying (7.4.1) using unidirectional search. The SLS uses the RCs estimated from 100 embeddings of the cover image as explained in Section 7.2.3, and linearly interpolates between grid points to cover the entire range of possible payloads.

The MDS makes use of the same logistic model as in (7.5.1), fit to each RC. A projected gradient descent with momentum initialized with IMS payloads for each bag was used to search for the payloads that minimize the deflection (7.4.5). To facilitate convergence, the learning rate and momentum were updated according to a one-cycle scheduler [208]; the learning rate and momentum fluctuated within the intervals $[10^{-2}, 10^{2}]$ and $[.90, .99]$, respectively. To comply with the payload constraint and bounds $0 \leq \alpha_i \leq \alpha_{\max}(\mathbf{x}^{(i)})$, at each step of the gradient descent the vector of payloads was projected to the feasible set of points, a hyperplane formed by (7.3.1) contained within the B-dimensional box $[0, \alpha_{\max}(\mathbf{x}^{(1)})] \times \ldots \times [0, \alpha_{\max}(\mathbf{x}^{(B)})]$.

7.6 Experiments

This section contains the results of all our experiments and their discussion. In particular, the proposed detector-informed senders are evaluated against the IMS with four pooling strategies.

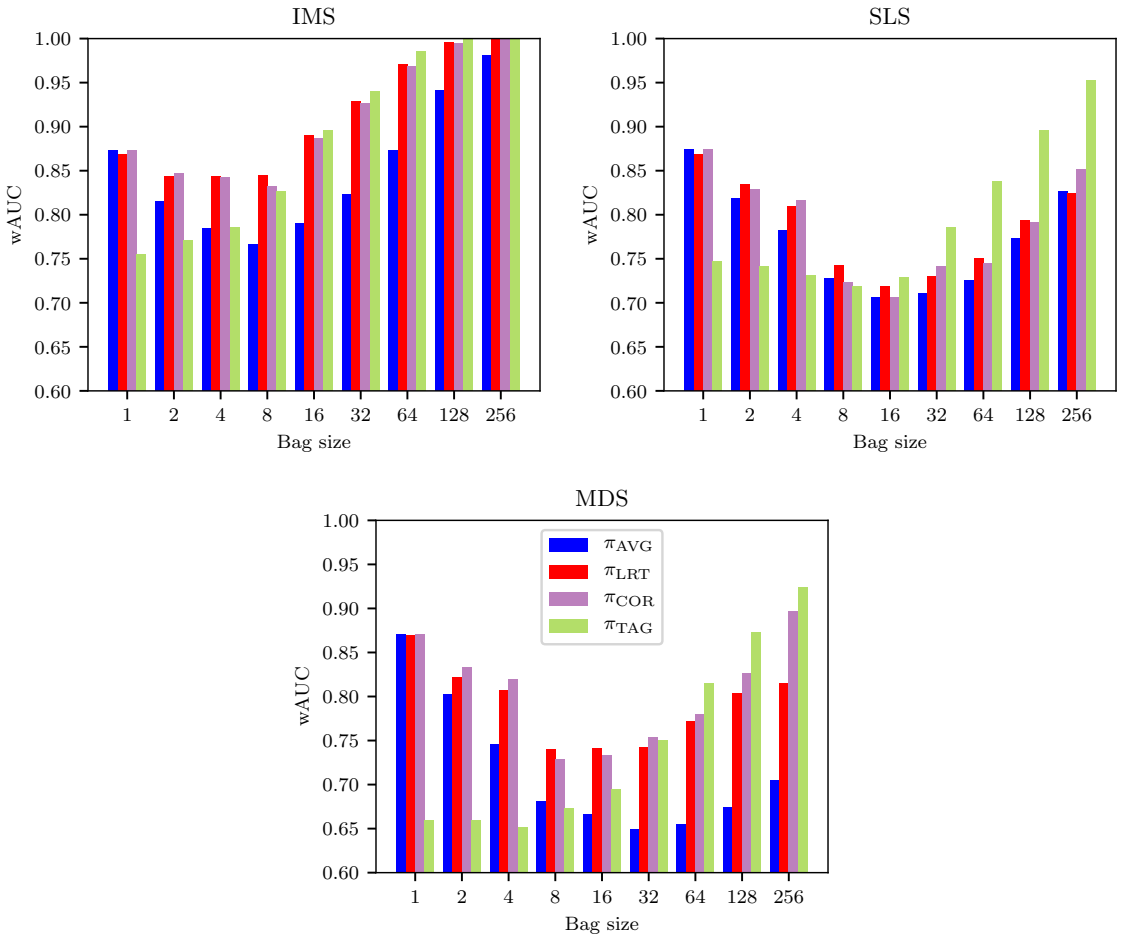


Figure 7.5.1: Detection accuracy of Warden's SRNet2 in terms of wAUC versus the bag size for IMS (top left), SLS (top right), and MDS (bottom) with four different pooling strategies.

Substantial space is devoted to studying the impact of the information available to the Warden as well as the effect of Warden's choices for the single-image detector.

7.6.1 Best spreading and pooling strategies

In this section, we compare the SLS and MDS and the previously proposed IMS. We also evaluate all poolers to see which pooling strategy is the best. We do so for a range of bags and one fixed setup with $r = 0.3$ bpp and HILL. The Warden uses the same architecture as the senders, the SRNet, trained on Split 2 (SRNet2) because it is not feasible to assume that the Warden has the same training set. In this section, we give the Warden the exact payloads $\alpha_{r,S}(\mathbf{C})$ that might be embedded in each bag. In reality, the Warden would have to estimate the payloads for each bag, which is likely to decrease the detectability. We simplify here because executing experiments at scale with having to estimate the payloads is very time consuming as the Warden needs to estimate the average response curves w.r.t. her detector for all images in the bag. In Section 7.6.2, we show that the effect of using the estimated payloads leads to only a small drop in detection accuracy and thus does not affect the results or our conclusions much.

The detection performance of pooled detectors is reported using the weighted Area Under the ROC Curve (wAUC) as used in ALASKA II [43]. We note that the pooled detector makes a binary decision about each bag being either cover or stego. Figure 7.5.1 shows the wAUC of four different poolers versus the bag size. Both detector-aware senders offer much better security when compared to the detector-agnostic IMS.

Note that for all senders, as the bag size grows, the detectability initially decreases and eventually starts increasing due to the Square Root Law since the senders maintain a positive communication rate r. The initial drop, which is far more pronounced for the two detector-aware senders, can be explained by considering the response curves. If a bag contains an image with a nearly flat response curve, it will be embedded close to its maximum capacity while other images will receive smaller payloads. Taking a bag of two as an example, it is more advantageous for the sender to embed payload 0.6 bpp in one of the images rather than 0.3 in each. The spreading thus initially helps decrease detectability to a point when the SRL starts engaging and the bags provide more data to reach a more reliable decision about the use of steganography. Note that this result is in stark contrast with the behavior of the IMS from [32] because the IMS there worked with fixed tags attached to all images and only embedded a given relative payload in each bag *on average*. Thus,

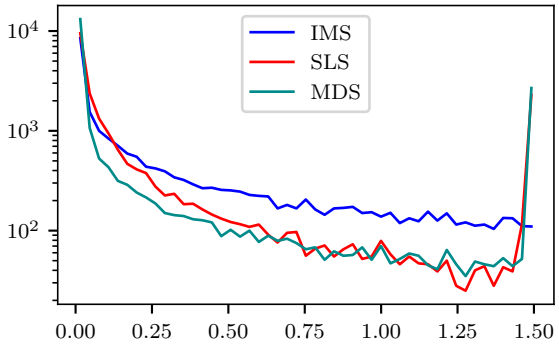


Figure 7.6.1: Distribution of relative payloads across the training dataset for IMS, SLS, and MDS for bag size $B = 100$. Note that the new detector-informed senders are far more aggressive in assigning payloads to images with most images either being embedded with small payloads and a significant fraction embedded fully.

it was unable to utilize the effect discussed above. Our concept of batch steganography in bags is more flexible and makes better use of the available cover images especially for small bags.

Continuing our discussion of Figure 7.5.1, we now comment on which poolers are the most effective in detecting batch steganography across the same range of bag sizes and for all three senders. For large bags, the best detection is obtained with the tag-based detector across all three senders because it is trained on the closest stego source. The correlator π_{COR} and the LRT π_{LRT} typically provide similar performance and are significantly better than the simple average π_{AVG}. This difference is most striking for the MDS because the simple average is essentially a correlator with uniform payloads. Thus, the more non-uniform the payload distribution is the larger the difference (see, e.g., the performance of π_{AVG} versus π_{COR} across the senders).

The poor performance of the tag-based pooler for small bag sizes is understandable because, as a binary detector on stego images embedded with tags, it performs poorly (and is also more difficult to train) as less than 14% of images have payload larger than 0.05 bpp with a high number of images with extremely small payloads. It starts being effective only for larger bag sizes, which are more likely to contain almost fully embedded images.

In Figure 7.6.1, we display the histogram of payloads embedded in images from the training set for all three senders, $B = 100$, and $r = 0.3$ bpp. The SLS and MDS are clearly much more aggressive in using certain images close to their maximal embedding capacity than the IMS. This is because these senders are aware of the fact that the embedding is "invisible" to the sender's SRNet. Understandably, this leads to a large gain in security at least as long as the Warden uses the same type of single-image detector. If the Warden uses a different detector for pooled steganalysis, the

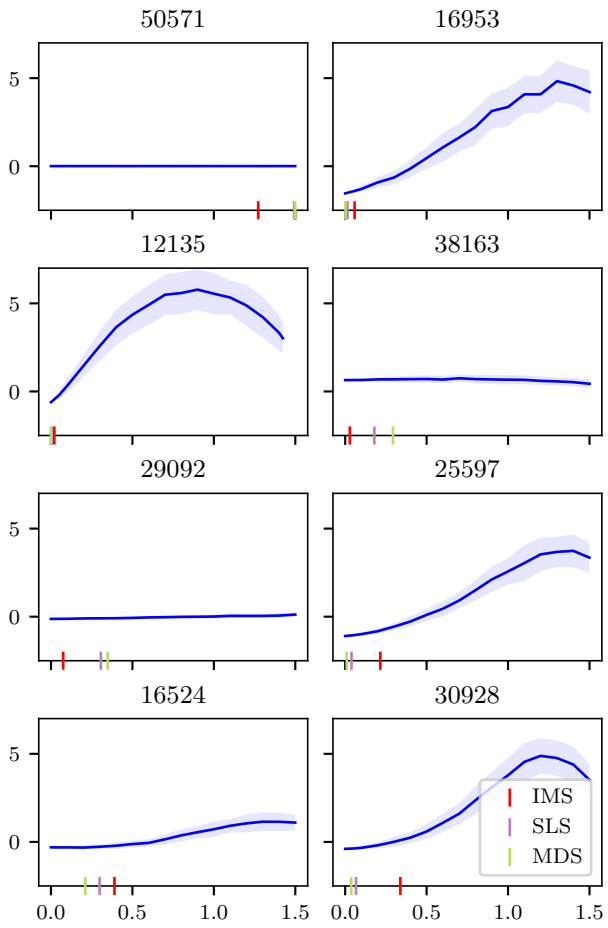


Figure 7.6.2: SRNet's response curves for a bag of 8 images with payloads allocated to each image by IMS, SLS, and MDS marked on the x-axis.

almost fully embedded images may become detectable if their response curves are not as flat as the sender's. We take a look at this important aspect in Section 7.6.3.

Figure 7.6.1 also shows that MDS is slightly more aggressive than SLS in allocating very large or very small payloads. This can be understood from Eqs. (7.4.3) and (7.4.7) showing the payloads as functions of the RC slopes. The payload of the MDS is inversely proportional to the square of the slope, making this sender more agressive when allocating the payload than the SLS. Figure 7.6.2 compares the three senders IMS, SLS, and MDS for a given bag of 8 images. For images 50571, 38163, and 29092, which have a flat response curve, the detector-aware senders embed larger payloads than the detector-agnostic IMS. For images with an increasing RC, such as 30928 and 25597, SLS and MDS are more conservative than IMS and allocate a smaller payload.

As the last experiment of this section, we include a study of the effect of the average communication

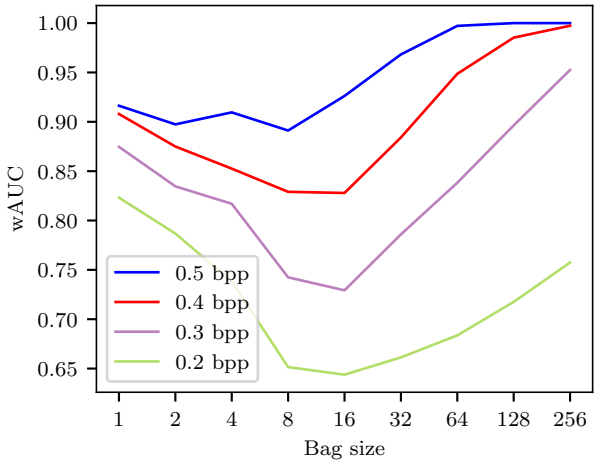


Figure 7.6.3: Detection accuracy of the best pooler of SRNet2 for SLS versus the bag size B and four communication rates r.

	π_{LRT}		π_{COR}	
B	16	64	16	64
SLS	.719 / .718	.750 / .746	.706 / .707	.745 / .756
MDS	.742 / .735	.771 / .757	.733 / .737	.780 / .768

Table 7.1: Accuracy (wAUC) of Warden's detectors for two senders, two bag sizes, and two pooling strategies with exact / estimated payloads. Warden's single-image detector is SRNet2, HILL 0.3 bpp.

rate r on the optimal bag size. We limit our study to the SLS and SRNet2 as Warden's detector. Figure 7.6.3 shows wAUC of the best pooler as a function of the bag size for four rate r. Note that with increased rate the dip becomes shallower and also starts moving towards smaller bag sizes.

7.6.2 Effect of estimating the payloads

In any realistic scenario, the Warden may know the algorithms used to embed and spread the payloads but not Alice's data. All three senders compute the payload size to be embedded in each image from the cover image itself. The Warden, however, will need to estimate the payloads from the images at hand. The embedding changes themselves may skew the estimated payload size should the Warden estimate from a stego image. For the IMS, the effect of the embedding changes on computing the embedding costs (or Fisher information for model-based steganography) is known to be practically negligible [78; 209]. For the new detector-aware senders, however, the payloads are also determined from the cover response curves, which are more sensitive to the embedding itself. For an image that receives a large payload, the Warden may end up with a very different response

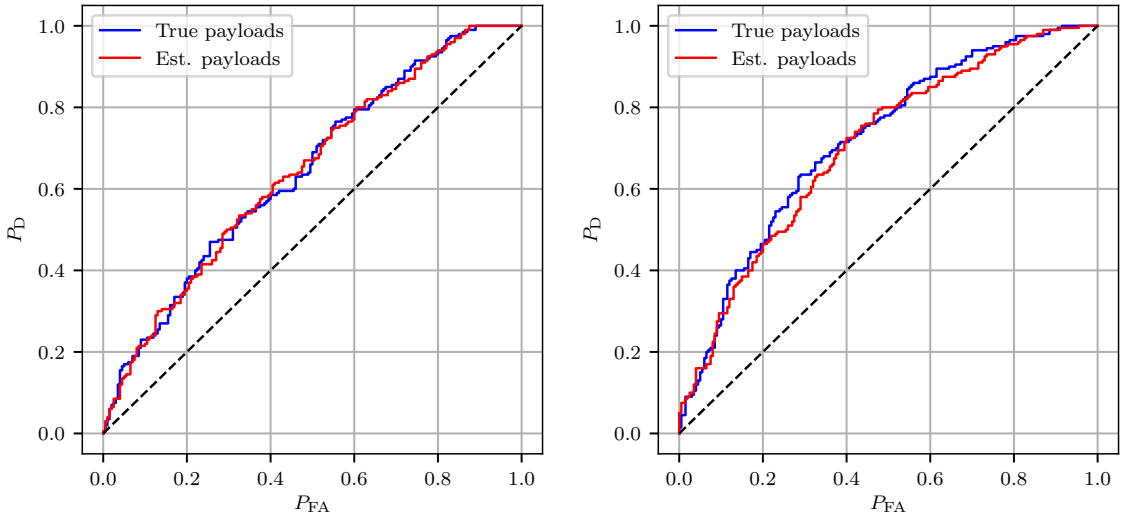


Figure 7.6.4: ROCs of pooling strategies using Warden's SRNet2 with exact and estimated payloads; π_{LRT} for SLS with bag size 16 (top) and π_{COR} for MDS with bag size 64 (bottom) for HILL at $r = 0.3$ bpp.

curve. Thus, even if she knows the spreading strategy, the communication rate, and the type of the detector used by the senders, the payloads that potentially reside in the images will be subject to an estimation error and lower the detection accuracy. We study this effect in this section.

First, it is hard to imagine that it would be advantageous for the Warden to intentionally mismatch the payloads potentially embedded in the images. Thus, the Warden should estimate them using a detector that is as close to the senders' detector as possible. As our first experiment, in Table 7.1 we compare the accuracy of the pooled detectors for a Warden who trains

1. SRNet2 on her dataset for d^{W} but uses the knowledge of the exact payloads $\alpha_{r,S}(\mathbf{C})$.

2. SRNet2 on her dataset for d^{W} and uses SRNet2 for estimating the payloads from the images at hand $\alpha_{r,S}(\mathbf{Y})$.

Note that Case 1 corresponds to the setup assumed in the previous section. In Figure 7.6.4, we show the ROCs corresponding to two selected entries of Table 7.1. While estimating the payloads leads to a performance drop, the effect is minimal because most images in the bag hold small payloads and thus their response curves are close to the response curves of the corresponding covers. For images embedded with medium to large payloads, which however form a small portion of each bag, the estimated payloads may be very different. Figure 7.6.5 shows the the relative payloads used by the sender as determined from her version of SRNet1 versus payloads estimated using SRNet2 by

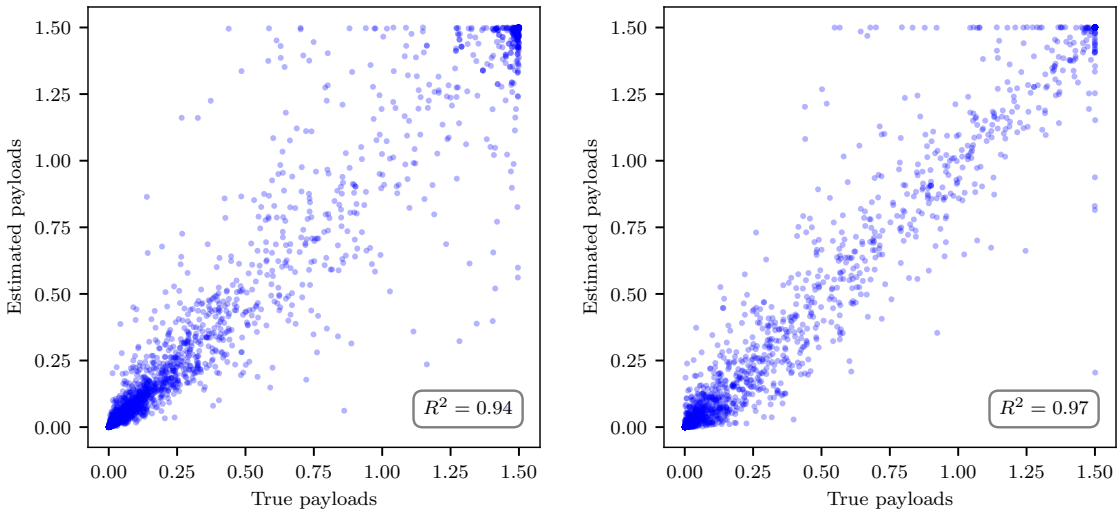


Figure 7.6.5: Payloads estimated by the Warden using SRNet2 versus the true embedded payloads as determined by the senders using SRNet1 for the SLS (top) and MDS (bottom). Bag size 16, HILL, $r = 0.3$ bpp.

the Warden from a HILL stego bag for SLS and MDS for $B = 16$ and $r = 0.3$ bpp.

7.6.3 Devious Warden

Since the SLS and MDS use feedback from a detector, while being more powerful than IMS when the Warden uses the same type of detector for pooling, they could potentially become vulnerable when the Warden intentionally or unintentionally mismatches the single-image detector. In this section, we study such a devious Warden who trains a different architecture (or a completely different single-image detector) on her training set. Since the effect of using payloads estimated from the images at hand instead of exact ones is small, we give the Warden the exact same payloads for pooling. This has been adopted for simplicity due to excessive computational cost of having to estimate the average response curves. Moreover, it helps us isolate the effect of the mismatched detector for pooling. The experiments were carried out for the SLS, MDS and IMS with SRNet2, EfN B4, Xu2, and SRM for a range of bag sizes. The results displayed in Table (7.2) show that the Warden indeed may gain from mismatching the detector. The gain is, however, quite small, and the detector-aware senders still exhibit a much better security than the IMS. In Figure (7.6.6), we show wAUC of Warden's best detector from among 16 different possibilities (four pooling strategies and four single-image detectors) as a function of the bag size. The new spreading strategies perform significantly better than IMS, even when considering different CNN architectures, training sets, and a very different

B / π	16 / π_{LRT}				64 / π_{TAG}			
	SRNet2	Xu2	B4	SRM	SRNet2	Xu2	B4	SRM
SLS	.7190	**.7324**	.7209	.6799	.8382	.7973	**.8567**	.7127
MDS	**.7416**	.7259	.7393	.7030	.8150	.7806	**.8421**	.7166
IMS	**.8902**	.8836	.8877	.6664	.9858	**.9907**	.9842	.7244

Table 7.2: Accuracy (wAUC) of Warden's detectors for three senders, two bag sizes, with two pooling strategies for HILL 0.3 bpp.

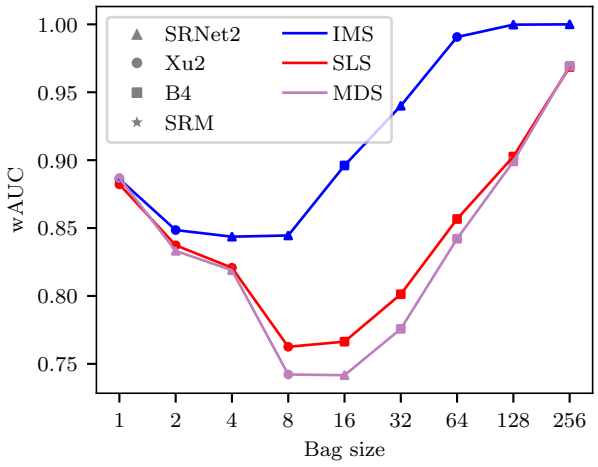


Figure 7.6.6: Accuracy (wAUC) of the best detector and best pooling strategy versus the bag size for IMS, SLS, and MDS. The best detector for each setting is highlighted using a different marker.

detector (SRM) than what Alice uses.

7.7 Conclusions

When communicating using steganography, the sender can be clever and choose to split the desired secret message among a bag of cover images to avoid being detected. In this chapter, we determine the sizes of the payload chunks by inspecting how each image in the bag reacts to embedding in terms of changing the soft output of a steganography detector as a function of the payload size, the image's "response curve." Two such detector-informed senders are investigated for spatial-domain steganography: 1) a sender that makes sure that all images in the bag experience the same shift in the detector response and 2) a sender that minimizes the sum of squares of the shifts, which can be interpreted as a deflection coefficient for a binary test distinguishing stego images from covers naturally corrupted by acquisition noise.

Using feedback from a detector indeed brings substantial improvement over the previously proposed image-merging sender that considers the bag as a single large image. The detectability as a function of the bag size for a fixed secret communication rate initially decreases, because the sender makes better use of all available covers, and then starts increasing due to the square root law since a fixed rate is maintained. We experimentally determined that the optimal bag size is 8–16 images per bag depending on the average communication rate.

On the detection side, we study three different strategies for the Warden to pool the outputs of her single-image detector: 1) correlator of the outputs with the expected detector output increase, 2) likelihood ratio test based on actual models of the detector output, and 3) detector trained on payload tags that the images would receive for sufficiently large bags. The likelihood ratio was the best pooling strategy for small to moderate bag sizes up to 16 while the tag based detector performed better for bag sizes larger than 16.

Using feedback from a detector for spreading can potentially backfire as the Warden may use a different detector for pooling. We looked into this issue in great detail by training alternative deep learning architectures as well as older rich-model based detectors. We discovered that doing so increases the Warden's accuracy, but not substantially and the detector-aware senders are still much more secure than the IMS.

In the future, we intend to further investigate the problem of optimal bag size by modeling the statistical collection of response curves. We also intend to explore the JPEG domain.

Chapter 8

Conclusion

Steganography and steganalysis of digital images were always considered to be vastly different from computer vision tasks. For this reason, many special ingredients were introduced and believed to be crucial in the success of deep learning methods in steganography and steganalysis. While deep learning models trained with these special ingredients outperformed the previous generation of feature-based statistical models, this dissertation proves that carefully getting rid of the special ingredients makes these models even more powerful. This dissertation also showcases some of the opportunities good steganography detectors bring when used as feedback for designing new steganographic schemes.

In the first part of this dissertation, I describe some of the challenges faced by deep learning in steganography and steganalysis and the solutions I proposed. First I describe the intriguing failure of convolutional neural networks in JPEG steganalysis due to their inability to build simple statistics in the DCT domain. The OneHotConv layer introduced to solve this issue turned out to be a great general purpose layer for modeling DCT coefficients for steganalysis and image forensics. Next, I tackle the difficulty of training models for each JPEG quality coefficient, I show that models can be trained on carefully designed ranges of JPEG quality factors without any loss of accuracy or robustness to perturbations in the quantization tables. This drastically reduces the computational cost of practical steganalysis.

In the next part I describe how my team and I used standard off-the-shelf computer vision convolutional neural network architectures in the ALASKA II steganalysis challenge. These networks were pre-trained on the large image classification dataset ImageNet, and fine-tuned for steganalysis with

very little domain tricks. Next I show that some simple targeted modifications to the architectures of these pre-trained models can increase their performance even more, these modifications aim at preserving high resolution feature maps in the first layers which improves their capacity to model the image noise.

In the last part of this dissertation, I turn to extracting human-interpretable feedback as to how state-of-the-art convolutional neural networks reach their decision. The folklore has it that, unlike older feature-based methods, which rely on a global accumulation of local handcrafted statistics, convolutional neural network can leverage spatially localized signals. I adapt existing interpretability tools to show that convolutional neural networks can indeed find overwhelming evidence for steganography from a few highly localized embedding artifacts. This constitutes the first opportunity for steganographers in designing future steganographic algorithms using human-interpretable feedback from trained detectors. Next I show that using an algorithmic feedback from state-of-the-art steganography detectors can give substantial gains in the context of batch steganography, where the sender uses a trained detector to determine an optimal way to spread the payload among a bag of images.

The findings presented in this dissertation hopefully pave the way for new avenues in steganography and steganalysis where deep learning is used to train high quality steganography detectors, and also serves as a companion for the steganographer to design better steganography, just as older feature-based methods did.

Bibliography

[1] Y. Yousfi and J. Fridrich, "An intriguing struggle of cnns in jpeg steganalysis and the onehot solution," *IEEE Signal Processing Letters*, vol. 27, pp. 830–834, 2020.

[2] Y. Yousfi, J. Butora, E. Khvedchenya, and J. Fridrich, "Imagenet pre-trained cnns for jpeg steganalysis," in *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[3] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2009.

[4] G. J. Simmons, "The prisoner's problem and the subliminal channel," in *Advances in Cryptology, CRYPTO '83*, D. Chaum, Ed. Santa Barbara, CA: New York: Plenum Press, August 22–24, 1983, pp. 51–67.

[5] N. Johnson, "Iot forensic considerations and steganography beyond images. invited talk," in *Network and Cloud Forensics Workshop, IEEE Conference on Communications and Network Security*, Las Vegas, Nevada, USA, October9–11, 2017.

[6] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color and gray-scale images," *IEEE Multimedia, Special Issue on Security*, vol. 8, no. 4, pp. 22–28, October–December 2001.

[7] S. Dumitrescu and X. Wu, "LSB steganalysis based on higher-order statistics," in *Proceedings of the 7th ACM Multimedia & Security Workshop*, A. M. Eskicioglu, J. Fridrich, and J. Dittmann, Eds., New York, NY, August 1–2, 2005, pp. 25–32.

[8] A. D. Ker, "A general framework for structural analysis of LSB replacement," in *Information Hiding, 7th International Workshop*, ser. Lecture Notes in Computer Science, M. Barni, J. Herrera, S. Katzenbeisser, and F. Pérez-González, Eds., vol. 3727. Barcelona, Spain: Springer-Verlag, Berlin, June 6–8, 2005, pp. 296–311.

[9] A. D. Ker and R. Böhme, "Revisiting weighted stego-image steganalysis," in *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, Eds., vol. 6819, San Jose, CA, January 27–31, 2008, pp. 5 1–17.

[10] J. Fridrich and J. Kodovský, "Steganalysis of LSB replacement using parity-aware features," in *Information Hiding, 14th International Conference*, ser. Lecture Notes in Computer Science, M. Kirchner and D. Ghosal, Eds., vol. 7692, Berkeley, California, May 15–18, 2012, pp. 31–45.

[11] R. Cogranne and F. Retraint, "Application of hypothesis testing theory for optimal detection of LSB matching data hiding," *Signal Processing*, vol. 93, no. 7, pp. 1724–1737, July, 2013.

[12] T. Denemark and J. Fridrich, "Detection of content-adaptive LSB matching (game theory

approach)," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2014*, A. Alattar, N. D. Memon, and C. Heitzenrater, Eds., vol. 9028, San Francisco, CA, February 3–5, 2014, pp. 04 1–12.

[13] A. D. Ker, "Steganalysis of LSB matching in grayscale images," *IEEE Signal Processing Letters*, vol. 12, no. 6, pp. 441–444, June 2005.

[14] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920–935, September 2011.

[15] M. Boroumand and J. Fridrich, "Synchronizing embedding changes in side-informed steganography," in *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2020*, San Francisco, CA, January 26–30 2020.

[16] T. Denemark and J. Fridrich, "Improving steganographic security by synchronizing the selection channel," in *3rd ACM IH&MMSec. Workshop*, J. Fridrich, P. Comesana, and A. Alattar, Eds., Portland, Oregon, June 17–19, 2015.

[17] Q. Giboulot, R. Cogranne, and P. Bas, "Synchronization Minimizing Statistical Detectability for Side-Informed JPEG Steganography," in *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03006635

[18] V. Holub and J. Fridrich, "Designing steganographic distortion using directional filters," in *Fourth IEEE International Workshop on Information Forensics and Security*, Tenerife, Spain, December 2–5, 2012.

[19] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion design for steganography in an arbitrary domain," *EURASIP Journal on Information Security, Special Issue on Revised Selected Papers of the 1st ACM IH and MMS Workshop*, vol. 2014:1, 2014.

[20] B. Li, M. Wang, and J. Huang, "A new cost function for spatial image steganography," in *Proceedings IEEE, International Conference on Image Processing, ICIP*, Paris, France, October 27–30, 2014.

[21] L. Guo, J. Ni, and Y. Q. Shi, "Uniform embedding for efficient JPEG steganography," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 5, pp. 814–825, May 2014.

[22] L. Guo, J. Ni, W. Su, C. Tang, and Y. Shi, "Using statistical image model for JPEG steganography: Uniform embedding revisited," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2669–2680, 2015.

[23] J. Fridrich and J. Kodovský, "Multivariate Gaussian model for designing additive distortion for steganography," in *Proc. IEEE ICASSP*, Vancouver, BC, May 26–31, 2013.

[24] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 221–234, 2016.

[25] V. Sedighi, J. Fridrich, and R. Cogranne, "Content-adaptive pentary steganography using the multivariate generalized Gaussian cover model," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon, Eds., vol. 9409, San Francisco, CA, February 8–12, 2015.

[26] P. Sallee, "Model-based steganography," in *Digital Watermarking, 2nd International Work-*

shop, ser. Lecture Notes in Computer Science, T. Kalker, I. J. Cox, and Y. M. Ro, Eds., vol. 2939. Seoul, Korea: Springer-Verlag, New York, October 20–22, 2003, pp. 154–167.

[27] ——, "Model-based methods for steganography and steganalysis," *International Journal of Image Graphics*, vol. 5, no. 1, pp. 167–190, 2005.

[28] R. Cogranne, Q. Giboulot, and P. Bas, "Steganography by minimizing statistical detectability: The cases of jpeg and color images," in *The 8th ACM Workshop on Information Hiding and Multimedia Security*, C. Riess and F. Schirrmacher, Eds. Denver, CO: ACM Press, 2020.

[29] A. D. Ker, "Batch steganography and the threshold game," in *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, E. J. Delp and P. W. Wong, Eds., vol. 6505, San Jose, CA, January 29–February 1, 2007, pp. 04 1–13.

[30] ——, "Perturbation hiding and the batch steganography problem," in *Information Hiding, 10th International Workshop*, ser. Lecture Notes in Computer Science, K. Solanki, K. Sullivan, and U. Madhow, Eds., vol. 5284. Santa Barbara, CA: Springer-Verlag, New York, June 19–21, 2008, pp. 45–59.

[31] A. D. Ker and T. Pevný, "Batch steganography in the real world," in *Proceedings of the 14th ACM Multimedia & Security Workshop*, J. Dittmann, S. Craver, and S. Katzenbeisser, Eds., Coventry, UK, September 6–7, 2012, pp. 1–10.

[32] V. Sedighi, R. Cogranne, and J. Fridrich, "Practical strategies for content-adaptive batch steganography and pooled steganalysis," in *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, March 5–9, 2017.

[33] M. Sharifzadeh, M. Aloraini, and D. Schonfeld, "Adaptive batch size image merging steganography and quantized Gaussian image steganography," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 867–879, 2020.

[34] L. Li, W. Zhang, C. Qin, K. Chen, W. Zhou, and N. Yu, "Adversarial batch image steganography against CNN-based pooled steganalysis," *Signal Processing*, vol. 181, pp. 107 920–107 920, 2021.

[35] A. D. Ker, "Locally square distortion and batch steganographic capacity," *International Journal of Digital Crime and Forensics*, vol. 1, no. 1, pp. 29–44, 2009.

[36] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Information Hiding, 3rd International Workshop*, ser. Lecture Notes in Computer Science, A. Pfitzmann, Ed., vol. 1768. Dresden, Germany: Springer-Verlag, New York, September 29–October 1, 1999, pp. 61–75.

[37] R. Cogranne, C. Zitzmann, L. Fillatre, F. Retraint, I. Nikiforov, and P. Cornu, "A cover image model for reliable steganalysis," in *Information Hiding, 13th International Conference*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, A. Ker, and S. Craver, Eds., Prague, Czech Republic, May 18–20, 2011, pp. 178–192.

[38] R. Cogranne, F. Retraint, C. Zitzmann, I. Nikiforov, L. Fillatre, and P. Cornu, "Hidden information detection using decision theory and quantized samples: Methodology, difficulties and results," *Digital Signal Processing*, vol. 24, pp. 144–161, 2014.

[39] R. Cogranne and F. Retraint, "An asymptotically uniformly most powerful test for LSB Matching detection," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 464–476, 2013.

[40] R. Cogranne, "Selection-channel-aware reverse JPEG compatibility for highly reliable steganalysis of JPEG images," in *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 4–8, 2020, pp. 2772–2776.

[41] J. Butora and J. Fridrich, "Reverse JPEG compatibility attack," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1444–1454, 2020.

[42] R. Cogranne, Q. Giboulot, and P. Bas, "The ALASKA steganalysis challenge: A first step towards steganalysis "Into the wild"," in *The 7th ACM Workshop on Information Hiding and Multimedia Security*, R. Cogranne and L. Verdoliva, Eds. Paris, France: ACM Press, July 3–5, 2019.

[43] R. Cogranne, Q. Giboulot, and P. Bas, "ALASKA–2: Challenging academic research on steganalysis with realistic images," in *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[44] Q. Giboulot, R. Cogranne, D. Borghys, and P. Bas, "Effects and Solutions of Cover-Source Mismatch in Image Steganalysis," *Signal Processing: Image Communication*, Aug. 2020.

[45] A. D. Ker and T. Pevnyý, "A mishmash of methods for mitigating the model mismatch mess," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2014*, A. Alattar, N. D. Memon, and C. Heitzenrater, Eds., vol. 9028, San Francisco, CA, February 3–5, 2014, pp. 1601–1615.

[46] J. KodovskÃœ, V. Sedighi, and J. Fridrich, "Study of cover source mismatch in steganalysis and ways to mitigate its impact," in *Media Watermarking, Security, and Forensics 2014*, A. M. Alattar, N. D. Memon, and C. D. Heitzenrater, Eds., vol. 9028, International Society for Optics and Photonics. SPIE, 2014, pp. 204 – 215.

[47] X. Zhang, X. Kong, P. Wang, and B. Wang, "Cover-source mismatch in deep spatial steganalysis," in *Digital Forensics and Watermarking*, H. W. X. Zhao, Y. Shi, H. J. Kim, and A. Piva, Eds. Springer International Publishing, 2020, pp. 71–83.

[48] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevný, "Moving steganography and steganalysis from the laboratory into the real world," in *1st ACM IH&MMSec. Workshop*, W. Puech, M. Chaumont, J. Dittmann, and P. Campisi, Eds., Montpellier, France, June 17–19, 2013.

[49] I. Lubenko and A. D. Ker, "Steganalysis with mismatched covers: Do simple classifiers help," in *Proc. 13th ACM Workshop on Multimedia and Security*, J. Dittmann, S. Katzenbeisser, and S. Craver, Eds., Coventry, UK, September 6–7 2012, pp. 11–18.

[50] ——, "Going from small to large data sets in steganalysis," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2012*, A. Alattar, N. D. Memon, and E. J. Delp, Eds., vol. 8303, San Francisco, CA, January 23–26, 2012, pp. OM 1–10.

[51] H. Farid, "Detecting steganographic messages in digital images," Dartmouth College, New Hampshire, Tech. Rep. TR2001-412, 2001.

[52] H. Farid and L. Siwei, "Detecting hidden messages using higher-order statistics and support vector machines," in *Information Hiding, 5th International Workshop*, ser. Lecture Notes in Computer Science, F. A. P. Petitcolas, Ed., vol. 2578. Noordwijkerhout, The Netherlands: Springer-Verlag, New York, October 7–9, 2002, pp. 340–354.

[53] S. Lyu and H. Farid, "Steganalysis using higher-order image statistics," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 1, pp. 111–119, 2006.

[54] I. Avcibas, N. D. Memon, and B. Sankur, "Steganalysis using image quality metrics," in *Proceedings SPIE, Electronic Imaging, Security and Watermarking of Multimedia Contents III*, E. J. Delp and P. W. Wong, Eds., vol. 4314, San Jose, CA, January 22–25, 2001, pp. 523–531.

[55] J. Fridrich, "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes," in *Information Hiding, 6th International Workshop*, ser. Lecture Notes in Computer Science, J. Fridrich, Ed., vol. 3200. Toronto, Canada: Springer-Verlag, New York, May 23–25, 2004, pp. 67–81.

[56] Y. Q. Shi, C. Chen, and W. Chen, "A Markov process based approach to effective attacking JPEG steganography," in *Information Hiding, 8th International Workshop*, ser. Lecture Notes in Computer Science, J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, Eds., vol. 4437. Alexandria, VA: Springer-Verlag, New York, July 10–12, 2006, pp. 249–264.

[57] T. Pevný and J. Fridrich, "Merging Markov and DCT features for multi-class JPEG steganalysis," in *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, E. J. Delp and P. W. Wong, Eds., vol. 6505, San Jose, CA, January 29–February 1, 2007, pp. 3 1–14.

[58] D. Zou, Y. Q. Shi, W. Su, and G. Xuan, "Steganalysis based on Markov model of thresholded prediction-error image," in *Proceedings IEEE, International Conference on Multimedia and Expo*, Toronto, Canada, July 9–12, 2006, pp. 1365–1368.

[59] T. Pevný, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," in *Proceedings of the 11th ACM Multimedia & Security Workshop*, J. Dittmann, S. Craver, and J. Fridrich, Eds., Princeton, NJ, September 7–8, 2009, pp. 75–84.

[60] ——, "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215–224, June 2010.

[61] J. Kodovský, T. Pevný, and J. Fridrich, "Modern steganalysis can detect YASS," in *Proceedings SPIE, Electronic Imaging, Media Forensics and Security II*, N. D. Memon, A. Alattar, E. J. Delp, and J. Dittmann, Eds., vol. 7541, San Jose, CA, January 17–21, 2010, pp. 02 01–11.

[62] T. Pevný, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *Information Hiding, 12th International Conference*, ser. Lecture Notes in Computer Science, R. Böhme and R. Safavi-Naini, Eds., vol. 6387. Calgary, Canada: Springer-Verlag, New York, June 28–30, 2010, pp. 161–177.

[63] J. Fridrich, J. Kodovský, M. Goljan, and V. Holub, "Steganalysis of content-adaptive steganography in spatial domain," in *Information Hiding, 13th International Conference*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, A. Ker, and S. Craver, Eds., Prague, Czech Republic, May 18–20, 2011, pp. 102–117.

[64] ——, "Breaking HUGO – the process discovery," in *Information Hiding, 13th International Conference*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, A. Ker, and S. Craver, Eds., Prague, Czech Republic, May 18–20, 2011, pp. 85–101.

[65] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, June 2011.

[66] J. Kodovský and J. Fridrich, "Steganalysis of JPEG images using rich models," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2012*, A. Alattar, N. D. Memon, and E. J. Delp, Eds., vol. 8303, San Francisco, CA, January 23–26, 2012, pp. 0A 1–13.

[67] Y. Q. Shi, P. Sutthiwan, and L. Chen, "Textural features for steganalysis," in *Information Hiding, 14th International Conference*, ser. Lecture Notes in Computer Science, M. Kirchner and D. Ghosal, Eds., vol. 7692, Berkeley, California, May 15–18, 2012, pp. 63–77.

[68] L. Chen, Y. Shi, P. Sutthiwan, and X. Niu, "A novel mapping scheme for steganalysis," in *International Workshop on Digital Forensics and Watermaking*, ser. LNCS, Y. Shi, H.-J. Kim, and F. Perez-Gonzalez, Eds., vol. 7809. Springer Berlin Heidelberg, 2013, pp. 19–33.

[69] L. Chen, Y.-Q. Shi, and P. Sutthiwan, "Variable multi-dimensional co-occurrence for steganalysis," in *Digital Forensics and Watermarking, 13th International Workshop, IWDW*, vol. 9023. Taipei, Taiwan: Springer, October 1–4 2014, pp. 559–573.

[70] J. Kodovský, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 432–444, April 2012.

[71] J. Kodovský and J. Fridrich, "Steganalysis in high dimensions: Fusing classifiers built on random subspaces," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics III*, A. Alattar, N. D. Memon, E. J. Delp, and J. Dittmann, Eds., vol. 7880, San Francisco, CA, January 23–26, 2011, pp. OL 1–13.

[72] R. Cogranne, T. Denemark, and J. Fridrich, "Optimal detection of steganography using a statistical model of fld ensemble classifiers," *submitted* 2014.

[73] R. Cogranne and J. Fridrich, "Modeling and extending the ensemble classifier for steganalysis of digital images using hypothesis testing theory," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 2627–2642, December 2015.

[74] M. Goljan, R. Cogranne, and J. Fridrich, "Rich model for steganalysis of color images," in *Sixth IEEE International Workshop on Information Forensics and Security*, Atlanta, GA, December 3–5, 2014.

[75] M. Goljan and J. Fridrich, "Cfa-aware features for steganalysis of color images," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon, Eds., vol. 9409, San Francisco, CA, February 8–12, 2015.

[76] H. Abdulrahman, M. Chaumont, P. Montesinos, and B. Magnier, "Color image steganalysis based on steerable gaussian filters bank," in *Proceedings of the 4th ACM workshop on Information Hiding and Multimedia Security*, 2016, pp. 109–114.

[77] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis against WOW embedding algorithm," in *2nd ACM IH&MMSec. Workshop*, A. Uhl, S. Katzenbeisser, R. Kwitt, and A. Piva, Eds., Salzburg, Austria, June 11–13, 2014, pp. 91–96.

[78] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, "Selection-channel-aware rich model for steganalysis of digital images," in *IEEE International Workshop on Information Forensics and Security*, Atlanta, GA, December 3–5, 2014.

[79] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis based on embedding probabilities of pixels," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 734–745, April 2016.

[80] T. Denemark, M. Boroumand, and J. Fridrich, "Steganalysis features for content-adaptive JPEG steganography," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1736–1746, August 2016.

[81] V. Holub and J. Fridrich, "Low-complexity features for JPEG steganalysis using undecimated DCT," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 219–228, February 2015.

[82] ——, "Phase-aware projection model for steganalysis of JPEG images," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon, Eds., vol. 9409, San Francisco, CA, February 8–12, 2015.

[83] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang, "Steganalysis of adaptive JPEG steganography using 2D Gabor filters," in *3rd ACM IH&MMSec. Workshop*, P. Comesana, J. Fridrich, and A. Alattar, Eds., Portland, Oregon, June 17–19, 2015.

[84] M. Boroumand and J. Fridrich, "Boosting steganalysis with explicit feature maps," in *4th ACM IH&MMSec. Workshop*, F. Perez-Gonzales, F. Cayre, and P. Bas, Eds., Vigo, Spain, June 20–22, 2016.

[85] J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG images: Breaking the F5 algorithm," in *Information Hiding, 5th International Workshop*, ser. Lecture Notes in Computer Science, vol. 2578. Noordwijkerhout, The Netherlands: Springer-Verlag, New York, October 7–9, 2002, pp. 310–323.

[86] J. Kodovský and J. Fridrich, "Calibration revisited," in *Proceedings of the 11th ACM Multimedia & Security Workshop*, J. Dittmann, S. Craver, and J. Fridrich, Eds., Princeton, NJ, September 7–8, 2009, pp. 63–74.

[87] R. Böhme, "Improved statistical steganalysis using models of heterogeneous cover signals," Ph.D. dissertation, Faculty of Computer Science, Technische Universität Dresden, Germany, 2008.

[88] R. Cogranne, V. Sedighi, T. Pevný, and J. Fridrich, "Is ensemble classifier needed for steganalysis in high-dimensional feature spaces?" in *IEEE International Workshop on Information Forensics and Security*, Rome, Italy, November 16–19, 2015.

[89] R. Cogranne, T. Denemark, and J. Fridrich, "Theoretical model of the FLD ensemble classifier based on hypothesis testing theory," in *IEEE International Workshop on Information Forensics and Security*, Atlanta, GA, USA, December 3–5 2014.

[90] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[91] S. Tan and B. Li, "Stacked convolutional auto-encoders for steganalysis of digital images," in *Signal and information processing association annual summit and conference (APSIPA), 2014 Asia-Pacific*. IEEE, 2014, pp. 1–4.

[92] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," in *Media Watermarking, Security, and Forensics 2015*, vol. 9409. SPIE, 2015, pp. 171–180.

[93] G. Xu, H. Z. Wu, and Y. Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, May 2016.

[94] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, November 2017.

[95] S. Wu, S.-H. Zhong, and Y. Liu, "Steganalysis via deep residual network," in *IEEE 22nd Inter-*

national Conference on Parallel and Distributed Systems (ICPADS), Wuhan, China, December 13–16, 2016.

[96] J. Yang, Y.-Q. Shi, E. Wong, and X. Kang, "JPEG steganalysis based on densenet," *CoRR*, vol. abs/1711.09335, 2017. [Online]. Available: http://arxiv.org/abs/1711.09335

[97] G. Xu, "Deep convolutional neural network to detect J-UNIWARD," in *The 5th ACM Workshop on Information Hiding and Multimedia Security*, M. Stamm, M. Kirchner, and S. Voloshynovskiy, Eds., Philadelphia, PA, June 20–22, 2017.

[98] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich, "JPEG-phase-aware convolutional neural network for steganalysis of JPEG images," in *The 5th ACM Workshop on Information Hiding and Multimedia Security*, M. Stamm, M. Kirchner, and S. Voloshynovskiy, Eds., Philadelphia, PA, June 20–22, 2017.

[99] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, May 2019.

[100] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-net: An efficient CNN for spatial steganalysis," in *IEEE ICASSP*, Alberta, Canada, April 15–20, 2018, pp. 2092–2096.

[101] J. Huang, J. Ni, L. Wan, and J. Yan, "A customized convolutional neural network with low model complexity for JPEG steganalysis." New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3335203.3335734

[102] X. Deng, B. Chen, W. Luo, and D. Luo, "Fast and effective global covariance pooling network for image steganalysis," in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, ser. IHMMSec'19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 230–234. [Online]. Available: https://doi.org/10.1145/3335203.3335739

[103] S. Tan, W. Wu, Z. Shao, Q. Li, B. Li, and J. Huang, "CALPA-NET: Channel-pruning-assisted deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 131–146, 2021.

[104] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," 2019.

[105] J. Yang, B. Lu, L. Xiao, X. Kang, and Y.-Q. Shi, "Reinforcement learning aided network architecture generation for JPEG image steganalysis," in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, ser. IHMMSec '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 23–32. [Online]. Available: https://doi.org/10.1145/3369412.3395060

[106] X. Deng, W. Luo, and Y. Fang, "Spatial steganalysis based on gradient-based neural architecture search," in *International Conference on Provable Security*. Springer, 2021, pp. 365–375.

[107] Y. Yousfi, J. Butora, Q. Giboulot, and J. Fridrich, "Breaking ALASKA: Color separation for steganalysis in JPEG domain," in *The 7th ACM Workshop on Information Hiding and Multimedia Security*, R. Cogranne and L. Verdoliva, Eds. Paris, France: ACM Press, July 3–5, 2019.

[108] J. Butora and J. Fridrich, "Detection of diversified stego sources using CNNs," in *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2019*, A. Alattar and N. D. Memon, Eds., San Francisco, CA, January 14–17, 2019.

[109] Y. Yousfi and J. Fridrich, "JPEG steganalysis detectors scalable with respect to compres-

sion quality," in *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2020*, A. Alattar and N. D. Memon, Eds., San Francisco, CA, 2020.

[110] C. Fuji-Tsang and J. Fridrich, "Steganalyzing images of arbitrary size with CNNs," in *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2018*, A. Alattar and N. D. Memon, Eds., San Francisco, CA, January 29–February 1, 2018.

[111] W. You, H. Zhang, and X. Zhao, "A siamese CNN for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 291–306, 2021.

[112] M. Yedroudj, M. Chaumont, F. Comby, A. Oulad Amara, and P. Bas, "Pixels-off: Data-augmentation complementary solution for deep-learning steganalysis," in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, ser. IHMMSec '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 39–48. [Online]. Available: https://doi.org/10.1145/3369412.3395061

[113] M. Yedroudj, M. Chaumont, and F. Comby, "How to augment a small learning set for improving the performances of a CNN-based steganalyzer?" in *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2018*, A. Alattar and N. D. Memon, Eds., San Francisco, CA, January 29–February 1, 2018.

[114] T. Itzhaki, Y. Yousfi, and J. Fridrich, "Data augmentation for JPEG steganalysis," in *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2021, pp. 1–6.

[115] I.-J. Yu, W. Ahn, S.-H. Nam, and H. Lee, "BitMix: data augmentation for image steganalysis," *Electronics Letters*, vol. 56, no. 24, pp. 1311–1314, November 2020. [Online]. Available: http://dx.doi.org/10.1049/el.2020.1951

[116] H. Ruiz, M. Chaumont, M. Yedroudj, A. O. Amara, F. Comby, and G. Subsol, "Analysis of the scalability of a deep-learning network for steganography "into the wild"," 2020.

[117] J. Fridrich, T. Pevný, and J. Kodovský, "Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities," in *Proceedings of the 9th ACM Multimedia & Security Workshop*, J. Dittmann and J. Fridrich, Eds., Dallas, TX, September 20–21, 2007, pp. 3–14.

[118] A. Westfeld, "F5 a steganographic algorithm: High capacity despite better steganalysis," in *4th International Workshop on Information Hiding*. Springer-Verlag, 2001, pp. 289–302.

[119] Y. Yousfi, J. Butora, J. Fridrich, and C. F. Tsang, "Improving EfficientNet for JPEG steganalysis," in *The 9th ACM Workshop on Information Hiding and Multimedia Security*, Brussels, Belgium, June 21–25, 2021, under review.

[120] J. Butora, Y. Yousfi, and J. Fridrich, "How to pretrain for steganalysis," in *The 9th ACM Workshop on Information Hiding and Multimedia Security*, D. Borghys and P. Bas, Eds. Brussels, Belgium: ACM Press, 2021.

[121] K. Chubachi, "An ensemble model using CNNs on different domains for ALASKA2 image steganalysis," in *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[122] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE conference on computer vision and pattern recognition*, June 20–25, 2009, pp. 248–255.

[123] D. Cozzolino, D. Gragnaniello, G. Poggi, and L. Verdoliva, "Towards universal GAN image detection," in *2021 International Conference on Visual Communications and Image Processing*

(VCIP). IEEE, 2021, pp. 1–5.

[124] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "FaceForensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[125] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer, "The deepfake detection challenge (DFDC) dataset," 2020. [Online]. Available: https://arxiv.org/abs/2006.07397

[126] A. Haliassos, K. Vougioukas, S. Petridis, and M. Pantic, "Lips don't lie: A generalisable and robust approach to face forgery detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5039–5049.

[127] Y. Yousfi, J. Butora, and J. Fridrich, "CNN steganalyzers leverage local embedding artifacts," in *IEEE International Workshop on Information Forensics and Security*, Montpellier, France, December 7–10, 2021.

[128] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, "Determining image origin and integrity using sensor noise," *IEEE Transactions on information forensics and security*, vol. 3, no. 1, pp. 74–90, 2008.

[129] J. Lukáš, J. Fridrich, and M. Goljan, "Determining digital image origin using sensor imperfections," in *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII*, E. J. Delp and P. W. Wong, Eds., San Jose, CA, January 16–20, 2005, pp. 249–260.

[130] M. Goljan, J. Fridrich, and T. Filler, "Large scale test of sensor fingerprint camera identification," in *Media forensics and security*, vol. 7254. SPIE, 2009, pp. 170–181.

[131] G. Healey and R. Kondepudy, "Radiometric CCD camera calibration and noise estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 267–276, 1994.

[132] A. Foi, "Clipped noisy images: Heteroskedastic modeling and practical denoising," *Signal Processing*, vol. 89, no. 12, pp. 2609–2629, 2009.

[133] T. H. Thai, R. Cogranne, and F. Retraint, "Camera model identification based on the heteroscedastic noise model," *Image Processing, IEEE Transactions on*, vol. 23, no. 1, pp. 250–263, Jan 2014.

[134] P. Bas, "Steganography via cover-source switching," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, December 4-7 2016, pp. 1–6.

[135] T. Taburet, P. Bas, W. Sawaya, and J. Fridrich, "Natural steganography in jpeg domain with a linear development pipeline," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 173–186, 2021.

[136] W. Pennebaker and J. Mitchell, *JPEG: Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.

[137] J. Butora and J. Fridrich, "Effect of jpeg quality on steganographic security," in *The 7th ACM Workshop on Information Hiding and Multimedia Security*, R. Cogranne and L. Verdoliva, Eds. Paris, France: ACM Press, July 3–5, 2019.

[138] ——, "Steganography and its detection in JPEG images obtained with the "trunc" quantizer,"

in *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 4–8, 2020.

[139] S. Agarwal and H. Farid, "Photo forensics from rounding artifacts," in *The 8th ACM Workshop on Information Hiding and Multimedia Security*, C. Riess and F. Schirrmacher, Eds. Denver, CO: ACM Press, 2020.

[140] ——, "Photo forensics from JPEG dimples," in *IEEE Workshop on Information Forensics and Security (WIFS)*, December 4-7, 2017.

[141] S. Mandelli, N. Bonettini, P. Bestagini, and S. Tubaro, "Training CNNs in presence of JPEG compression: Multimedia forensics vs computer vision," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2020, pp. 1–6.

[142] B. Lorch, A. Maier, and C. Riess, "Reliable JPEG forensics via model uncertainty," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2020, pp. 1–6.

[143] P. Bas, T. Filler, and T. Pevný, "Break our steganographic system – the ins and outs of organizing BOSS," in *Information Hiding, 13th International Conference*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, A. Ker, and S. Craver, Eds., vol. 6958, Prague, Czech Republic, May 18–20, 2011, pp. 59–70.

[144] P. Bas and T. Furon, "BOWS-2," http://bows2.ec-lille.fr, July 2007.

[145] J. Zeng, S. Tan, B. Li, and J. Huang, "Large-scale JPEG image steganalysis using hybrid deep-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1200–1214, May 2018.

[146] M. Chaumont, "Deep learning in steganography and steganalysis," in *Digital Media Steganography: Principles, Algorithms, Advances*, M. Hassaballah, Ed. Elsevier, 2020, ch. 14, pp. 321–349.

[147] Y. Yousfi, J. Fridrich, J. Butora, and Q. Giboulot, "Breaking ALASKA: Color separation for steganalysis in JPEG domain," in *The 7th ACM Workshop on Information Hiding and Multimedia Security*, R. Cogranne and L. Verdoliva, Eds. Paris, France: ACM Press, July 3–5, 2019.

[148] B. K. R. L. L. Gueguen, A. Sergeev and J. Yosinski, "Faster neural networks straight from JPEG," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 3933–3944. [Online]. Available: http://papers.nips.cc/paper/7649-faster-neural-networks-straight-from-jpeg.pdf

[149] M. Ehrlich and L. S. Davis, "Deep residual learning in the JPEG transform domain," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3484–3493.

[150] V. Sedighi and J. Fridrich, "Histogram layer, moving convolutional neural networks towards feature-based steganalysis," in *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2017*, A. Alattar and N. D. Memon, Eds., San Francisco, CA, January 29–February 1, 2017.

[151] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," *Wavelets, Time-Frequency Methods and Phase Space*, January 1989.

[152] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic

image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[153] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[154] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.75

[155] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *2015 IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.304

[156] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.579

[157] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *2nd International Conference on Learning Representations, (ICLR) 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, April 2014.

[158] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00781

[159] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," in *Advances in Neural Information Processing Systems*, 2018, pp. 9605–9616.

[160] C. Wang and J. Ni, "An efficient JPEG steganographic scheme based on the block–entropy of DCT coefficents," in *Proc. of IEEE ICASSP*, Kyoto, Japan, March 25–30, 2012.

[161] J. Kodovský, V. Sedighi, and J. Fridrich, "Study of cover source mismatch in steganalysis and ways to mitigate its impact," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2014*, A. Alattar, N. D. Memon, and C. Heitzenrater, Eds., San Francisco, CA, February 3–5, 2014.

[162] L. Zeng, X. Kong, M. Li, and Y. Guo, "JPEG quantization table mismatched steganalysis via robust discriminative feature transformation," in *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon, Eds., vol. 9409, San Francisco, CA, February 8–12, 2015.

[163] J. Pasquet, S. Bringay, and M. Chaumont, "Steganalysis with cover-source mismatch and a small learning database," in *2014 22nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2014, pp. 2425–2429.

[164] Q. Giboulot, R. Cogranne, and P. Bas, "Steganalysis into the wild: How to define a source?" January 29–February 1, 2018.

[165] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 27–30 2016, pp. 770–778.

[166] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014, http://arxiv.org/abs/1412.6980.

[167] J. Butora and J. Fridrich, "Reverse JPEG compatibility attack," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1444–1454, September 2019.

[168] S. Ozcan and A. F. Mustacoglu, "Transfer learning effects on image steganalysis with pretrained deep residual neural network model," in *IEEE International Conference on Big Data (Big Data)*, December 10–13, 2018, pp. 2280–2287.

[169] T. Ridnik, H. Lawen, A. Noy, and I. Friedman, "TResNet: High performance GPU-dedicated architecture," *arXiv preprint arXiv:2003.13630*, 2020.

[170] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, June 16–20, 2019, pp. 510–519.

[171] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, July 21–26, 2017.

[172] T. Mingxing and V. L. Quoc, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML*, vol. 97, June 9–15, 2019, pp. 6105–6114.

[173] ——, "MixConv: Mixed depthwise convolutional kernels," in *British Machine Vision Conference, BMVC*, September 9–12, 2019.

[174] S. Kornblith, J. Shlens, and Q. V. Le, "Do better ImageNet models transfer better?" in *Proceedings of the IEEE conference on computer vision and pattern recognition*, June 16–20, 2019, pp. 2661–2671.

[175] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[176] R. Cogranne, Q. Giboulot, and P. Bas, "Steganography by minimizing statistical detectability: The cases of JPEG and color images," in *The 8th ACM Workshop on Information Hiding and Multimedia Security*, C. Riess and F. Schirrmacher, Eds. Denver, CO: ACM Press, 2020.

[177] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[178] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[179] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, December 3–8, 2018, pp. 6638–6648.

[180] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM sigkdd international conference on knowledge discovery and data mining*, August 13–17, 2016, pp. 785–794.

[181] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "Faceforensics++:

Learning to detect manipulated facial images," in *2019 IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1–11.

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

[183] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva, "ForensicTransfer: Weakly-supervised domain adaptation for forgery detection," *arXiv*, 2018. [Online]. Available: http://arxiv.org/abs/1812.02510

[184] S. Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8692–8701.

[185] G. Xu, "1st place solution," https://www.kaggle.com/c/alaska2-image-steganalysis/discussion/168548, 2020, [Online; accessed 29-December-2020].

[186] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, 2015.

[187] S. Tan, W. Wu, Z. Shao, Q. Li, B. Li, and J. Huang, "CALPA-NET: Channel-pruning-assisted deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 131–146, 2021.

[188] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[189] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[190] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[191] Q. Ha, "Single effnet-b0 private LB 0.921: How to modify effnet architecture," https://www.kaggle.com/c/alaska2-image-steganalysis/discussion/168542, 2020, [Online; accessed 29-December-2020].

[192] R. Cogranne, "J-MiPOD source code," personal communication, [e-mail; sent 21-December-2020].

[193] R. Cogranne, V. Sedighi, and J. Fridrich, "Practical strategies for contentadaptive batch steganography and pooled steganalysis," in *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, March 5–9, 2017, pp. 2122–2126.

[194] T. Pevný and J. Fridrich, "Novelty detection in blind steganalysis," in *Proceedings of the 10th ACM Multimedia & Security Workshop*, A. D. Ker, J. Dittmann, and J. Fridrich, Eds., Oxford, UK, September 22–23, 2008, pp. 167–176.

[195] Y. Yousfi, J. Butora, J. Fridrich, and C. Fuji Tsang, "Improving efficientnet for JPEG steganalysis," in *The 9th ACM Workshop on Information Hiding and Multimedia Security*, June 22–25, 2021.

[196] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Inter-*

national conference on machine learning. PMLR, 2017, pp. 3319–3328.

[197] E. J. Friedman, "Paths and consistency in additive cost sharing," *International Journal of Game Theory*, vol. 32, no. 4, pp. 501–518, 2004.

[198] P. Sturmfels, S. Lundberg, and S.-I. Lee, "Visualizing the impact of feature attribution baselines," *Distill*, 2020, http://distill.pub/2020/attribution-baselines.

[199] D. Upham, "Steganographic algorithm JSteg," Software available at http://zooid.org/ paul/crypto/jsteg.

[200] A. D. Ker, "Batch steganography and pooled steganalysis," in *Information Hiding, 8th International Workshop*, ser. Lecture Notes in Computer Science, J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, Eds., vol. 4437. Alexandria, VA: Springer-Verlag, New York, July 10–12, 2006, pp. 265–281.

[201] A. D. Ker and T. Pevný, "The steganographer is the outlier: Realistic large-scale steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 9, pp. 1424–1435, September 2014.

[202] T. Pevný and I. Nikolaev, "Optimizing pooling function for pooled steganalysis," in *IEEE International Workshop on Information Forensics and Security*, Rome, Italy, November 16–19, 2015, pp. 1–6.

[203] W. Tang, B. Li, S. Tan, M. Barni, and J. Huang, "CNN-based adversarial embedding for image steganography," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2074–2087, 2019.

[204] X. Hu, J. Ni, W. Zhang, and J. Huang, "Efficient JPEG batch steganography using intrinsic energy of image contents," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4544–4558, 2021.

[205] J. Fridrich, M. Goljan, D. Soukal, and P. Lisoněk, "Writing on wet paper," in *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII*, E. J. Delp and P. W. Wong, Eds., vol. 5681, San Jose, CA, January 16–20, 2005, pp. 328–340.

[206] A. D. Ker, "The square root law of steganography," in *The 5th ACM Workshop on Information Hiding and Multimedia Security*, M. Stamm, M. Kirchner, and S. Voloshynovskiy, Eds. Philadelphia, PA: ACM Press, June 20–22, 2017.

[207] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume II: Detection Theory*. Upper Saddle River, NJ: Prentice Hall, 1998, vol. II.

[208] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.

[209] V. Sedighi and J. Fridrich, "Effect of imprecise knowledge of the selection channel on steganalysis," in *3rd ACM IH&MMSec. Workshop*, J. Fridrich, P. Comesana, and A. Alattar, Eds., Portland, Oregon, June 17–19, 2015.