JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images

Mo Chen, Vahid Sedighi, Mehdi Boroumand, and Jessica Fridrich
Binghamton University
Department of ECE
Binghamton, NY 13902
mochen8@gmail.com,{vsedigh1,mboroum1,fridrich}@binghamton.edu

ABSTRACT

Detection of modern JPEG steganographic algorithms has traditionally relied on features aware of the JPEG phase. In this paper, we port JPEG-phase awareness into the architecture of a convolutional neural network to boost the detection accuracy of such detectors. Another innovative concept introduced into the detector is the "catalyst kernel" that, together with traditional high-pass filters used to pre-process images allows the network to learn kernels more relevant for detection of stego signal introduced by JPEG steganography. Experiments with J-UNIWARD and UED-JC embedding algorithms are used to demonstrate the merit of the proposed design.

KEYWORDS

Steganography, steganalysis, convolutional neural network, JPEG, phase aware, catalyst kernel

ACM Reference format:
Mo Chen, Vahid Sedighi, Mehdi Boroumand, and Jessica Fridrich. 2017. JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images. In *Proceedings of IH&MMSec'17, Philadelphia, PA, USA, Jun 20-22, 2017,* 10 pages.
https://doi.org/http://dx.doi.org/10.1145/3082031.3083248

1 INTRODUCTION

Steganography in its modern form is a private, covert communication method in which the sender hides the message inside an innocuous looking cover object using an algorithm driven by a secret shared with the recipient. The communication channel is observed by an adversary or warden who tries to establish whether the communicating parties use steganography. While covers can have many different forms, the most popular and also practical choices for the steganographers are digital media files. Among them, the JPEG format is by far the most prevalent image format in current use due to its efficiency to compress images with a small loss of perceptual quality.

IH&MMSec'17, Jun 20-22, 2017, Philadelphia, PA, USA

ACM ISBN ISBN 978-1-4503-5061-7/17/06…$15.00.
https://doi.org/http://dx.doi.org/10.1145/3082031.3083248

The first steganographic algorithms designed for the JPEG format were Jsteg [26] and OutGuess [17], followed by the F5 [28], Steghide [11], and Model-Based Steganography [20]. Because Jsteg and OutGuess predictably modify the histogram of DCT coefficients, the early attacks were based on first-order statistics of Discrete Cosine Transform (DCT) coefficients [32]. Later, significantly more accurate detectors were constructed as likelihood ratio tests with the aid of better models of cover DCT coefficients [24, 25]. The best detectors for F5 and its improved version nsF5 [7] as well as for Model-Based Steganography and Steghide are currently constructed as classifiers trained on features formed from quantized DCT coefficients, the JPEG Rich Model (JRM) [15]. Modern schemes, such as J-UNIWARD [14] and variants of UED [9, 10] are currently best detected using the so-called JPEG-phase-aware features formed from residuals extracted in the spatial domain split by their JPEG phase. Examples of such features include DCTR [12], GFR [22], PHARM [13], and their selection-channel-aware versions [5]. The key concept in their design is the notion of JPEG phase, which is the location of the residual with respect to the JPEG 8×8 grid. By splitting the statistics collected from the residuals by their phase more accurate detectors can be built.

Recently, novel detector architectures implemented within the paradigm of Convolutional Neural Networks (CNN) have been proposed for steganalysis of spatial-domain steganography [18, 23, 29, 30]. In this paper, we adapt CNN-based detectors for detection of modern JPEG-domain steganography by porting the concept of JPEG-phase-awareness into the network architecture. To this end, we implement a new phase-split layer and study two ways for incorporating phase awareness within the network architecture. Additionally, we augment the KV kernel traditionally used to prefilter images for CNN detectors by a second fixed kernel that works as a "catalyst" and allows the network to learn kernels that are more suitable for detecting stego noise introduced by JPEG-domain embedding schemes. After initial experiments with the number of layers, kernels, the network depth and height, and forming ensembles of networks by bagging on the training set, we arrived at a design capable of improving upon the state-of-the-art selection-channel-aware Gabor Filter Residuals (GFR) [5].

In the next section, we briefly introduce what we call JPEG phase. In Section 3, we describe two possible network architectures capable of working with phase-aware feature maps, the PNet and the VNet, and explain our design choices. Section 4 contains the results of all experiments on two steganographic algorithms, J-UNIWARD and UED-JC across a range of payloads and two JPEG quality factors. The performance is reported for both PNets and VNets in their individual forms as well as ensembles over bagged training sets.

Additionally, we study the impact of both cover and stego source mismatch on the detectors' accuracy. The paper is concluded in Section 5 where we also outline our future effort.

2 JPEG PHASE

In this section, we briefly review the concept of JPEG phase as used in steganalysis. WLOG, we will assume that the image is grayscale with $n_1 \times n_2$ pixels with n_1, n_2 multiples of 8. In a decompressed JPEG, the statistical properties of the stego signal are not spatially invariant within each 8×8 block because they depend on the position of the pixel and its neighborhood within the JPEG 8×8 pixel grid. It thus makes sense to collect statistics of pixels separately for each phase (i, j), $0 \le i, j \le 7$, which is defined as a sublattice of pixels with indices

$$\mathcal{L}_{ij} = \{(i + 8k, j + 8l) | 0 \le k < n_1/8, 0 \le l < n_2/8\}. \quad (1)$$

The computation of JPEG-phase-aware features starts with computing noise residuals by convolving the decompressed (non-rounded) JPEG image $\mathbf{x} \in \mathbb{R}^{n_1 \times n_2}$ with kernels $\mathbf{g} \in \mathbb{R}^{k_1 \times k_2}$ from some filter bank $\mathcal{B}$, $\mathbf{z}(\mathbf{x}, \mathbf{g}) = \mathbf{x} \star \mathbf{g}$. For example, in GFR [22] the bank $\mathcal{B}$ is formed by discretized 2D Gabor bases

$$g_{\lambda,\theta,\phi}(x, y) = e^{-(x'^2+\gamma^2 y'^2)/2\sigma^2} \cos\left(2\pi \frac{x'}{\lambda} + \phi\right), \quad (2)$$

where $x' = x \cos\theta + y \sin\theta$, $y' = -x \sin\theta + y \cos\theta$, $\sigma = 0.56\lambda$, $\gamma = 0.5$. Next, $\mathbf{z}$ is quantized, $\mathbf{r} = Q_Q(\mathbf{z}/q)$, where $q > 0$ is a fixed quantization step and $Q = \{-T, \dots, -1, 0, 1, 2, \dots, T\}$ a set of $2T + 1$ bin centroids with a truncation threshold T. Each residual is used to compute 64 histograms for each JPEG phase $0 \le i, j \le 7$, $0 \le m \le T$:

$$h_m^{(i,j)}(\mathbf{x}, \mathbf{g}, Q) = \sum_{(a,b)\in\mathcal{L}_{ij}} [|r_{ab}(\mathbf{x}, \mathbf{g}, Q)| = m], \quad (3)$$

where $[P]$ is the Iverson bracket equal to 1 when the statement P is true and 0 otherwise. All $T + 1$ values, $h_0^{(i,j)}, \dots, h_T^{(i,j)}$ from each histogram are concatenated into a vector of $64 \times (T + 1)$ values and these vectors are then concatenated for all kernels $\mathbf{g} \in \mathcal{B}$. The resulting feature vector with $64 \times (T + 1) \times |\mathcal{B}|$ elements is finally symmetrized based on the symmetries of natural images and kernels $\mathbf{g}$. Depending on the filter bank, different versions of JPEG-phase-aware features can be obtained [12, 13, 22].

In [5], the authors showed how the knowledge of the selection channel (the embedding change probabilities) can be incorporated into phase-aware features in a manner that can be considered as a generalization of the principle used in the maxSRM feature vector [4] used for steganalysis in the spatial domain. The detectors proposed here will be evaluated against this selection-channel-aware version of GFR features abbreviated as SCA-GFR.

3 THE PROPOSED PHASE-AWARE CNN

In this section, we describe the overall architecture of the proposed JPEG-phase-aware CNN. All key design components are explained and justified through experiments.

Recently, Xu et al. [29, 30] introduced a five-layer CNN for spatial-domain steganalysis with competitive performance. Each layer in this network processes the feature maps outputted by the previous layer in four steps: convolution, batch normalization, non-linear activation, and pooling. This CNN structure design contains a few key elements that are important to CNN-based steganalysis. In the first convolutional layer, an absolute value activation (ABS) layer is used to facilitate and improve statistical modeling in the subsequent layers. The TanH activation is employed at the early stages to limit the range of data values and prevent the deeper layers from modeling statistically insignificant large values, while the batch normalization (BN) is applied before each non-linear activation to normalize the weak stego noise and improve the net convergence. The spatial size of convolutional kernels in deeper layers is 1×1 to limit the receptive filter size and reduce the support region of modeling. From now on, we refer to this architecture as XuNet.

In our investigation, we started with the XuNet and modified it to allow the network to process information for each JPEG phase separately, together with several other important modifications. We view the first two layers of XuNet (not counting the fixed kernel) as "feature extractors" (non-linear residuals), which are then in layers 3–5 combined and compacted into a low-dimensional feature vector used for detection.

To incorporate phase awareness into this architecture, we need to disable pooling in the first two layers because it would mix different phases. The phase-split layer is applied to feature maps outputted by the second layer by subsampling each feature map on 64 sublattices (1). The network than continues the computation on 64 eight-times smaller feature maps.

We investigated two possibilities for the network architecture behind the phase-split layer. In what we termed a 'PNet' (Fig. 1 left), after the phase split we let each JPEG phase go through its own independent channel. This is similar in spirit to how JPEG-phase-aware features are formed. Forcing the split, however, increases the number of channels from 16 to 1024 (by factor of 64) and the net becomes wider towards the last layers. This increases memory requirements and also slows down training and testing due to increased computational complexity.

As an alternative, we allowed the net itself to merge or split the channels, which resulted in an alternative architecture depicted in Fig. 1 right, the VNet. This version allows channel merging and is more compact and faster to train. It also allows the model to take advantage of the correlation between different phases in layers 3–5.

In a VNet, instead of being grouped the outputs of the PhaseSplit module, 64 $16 \times (64 \times 64)$ feature maps, are concatenated into $1024 \times (64 \times 64)$ feature maps and fully convolved in the convolutional layer in Group 3. This channel reduction speeds up training and testing. Both versions of the CNN are shown in Fig. 1 in their final forms, which will now be justified on experiments.

To determine the network architecture, we used a greedy approach in which we incrementally modified one layer of the CNN while keeping everything else fixed and tested it on a fixed experimental setup described next. We describe the details of this process for the PNet only due to space limitations.

3.1 Standard developer setup

This setup involves attacking J-UNIWARD [14] at payload 0.4 bpnzac (bits per non-zero AC DCT coefficient) on BOSSbase 1.01 [1] compressed with JPEG quality factor 75. This source was split

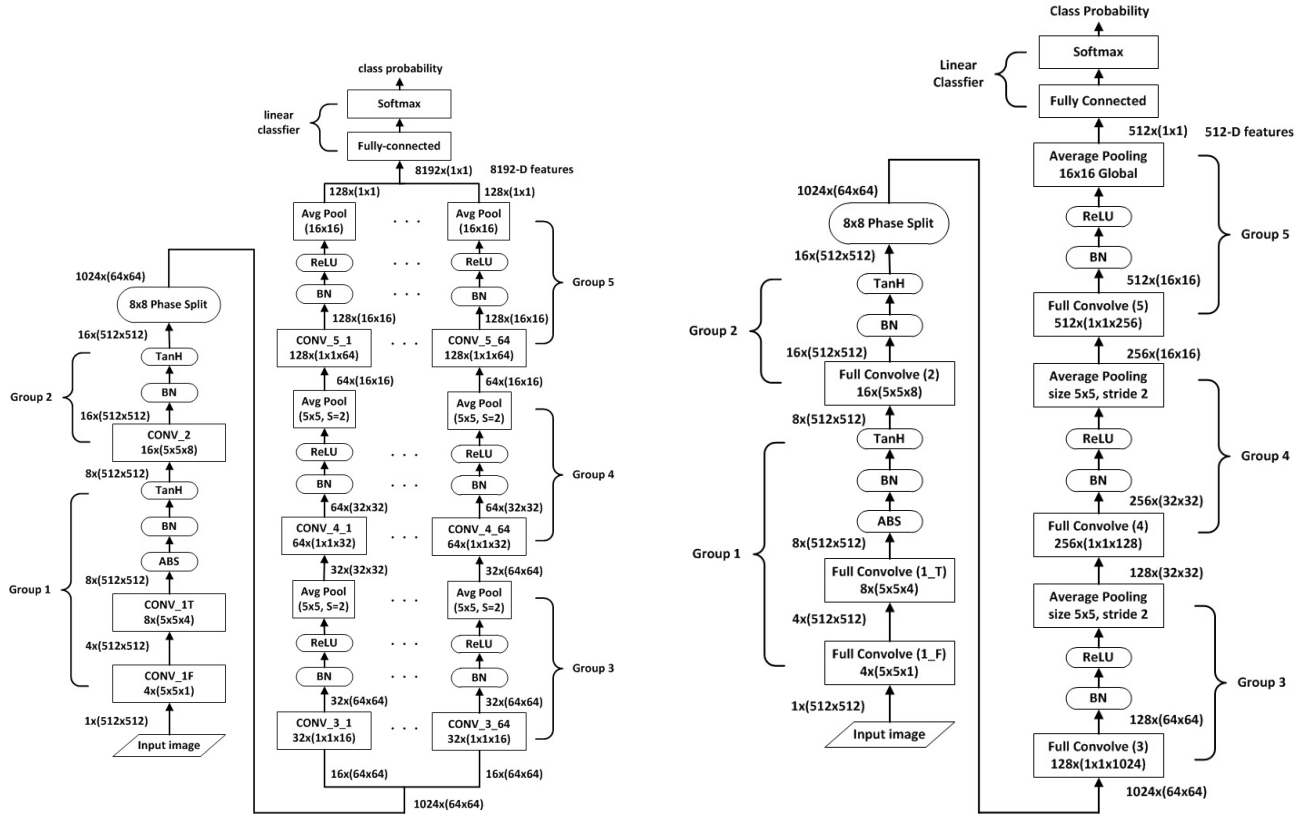


Figure 1: Two phase-aware CNN architectures: PNet (left) and VNet (right). Layer types and parameter sizes are inside the boxes. Data sizes are displayed on the sides. The dimensions of convolution kernels follow the format: (number of kernels) × (height × width × number of input feature maps) and the dimensions of data: (number of feature maps) × (height × width). Padding by 2 is applied in all convolutional layers and pooling layers except for the last pooling layer.

into a training and testing set by randomly selecting the images in a 60/40 ratio. The detection performance was measured with the total classification error probability under equal priors $P_E = \min_{P_{FA}} \frac{1}{2}(P_{FA} + P_{MD})$, where P_{FA} and P_{MD} are the false-alarm and missed-detection probabilities. Sometimes in this paper, we will present the error $P_E \in [0, 1]$ in percentage units, which is $P_E \times 100$. To assess the robustness of the detector, we also report the testing error on BOWS2 [2] to see the impact of testing on a different source than the source on which the detector was trained.

3.2 Overall architecture

The PNet depicted in Fig. 1 left consists of a convolutional module that transforms an input image into a "feature vector" of dimensionality 8192, and a linear classification module consisting of a fully-connected layer and a softmax layer outputting the probabilities for each class (cover vs. stego).

Similar to XuNet, the convolutional module has five groups of layers marked as 'Group 1' to 'Group 5' in Fig. 1. The CNN is also equipped with the TanH non-linear activation in Groups 1 and 2, and the Rectified Linear Unit (ReLU) in Groups 3–5. In Group 1, an absolute value activation (ABS) layer is used to force the statistical modeling to take into account the sign symmetry of noise residuals. Additionally, Batch Normalization (BN) is performed right before each non-linear activation layer to normalize each feature map to zero-mean and unit-variance, and thus help the gradient descend back-propagation algorithm avoid being trapped in local minima. The network employs TanH instead of ReLU in Groups 1 and 2 to limit the range of data values and prevent the deeper layers from modeling large values, as they are sparse and not statistically significant. In Groups 3–5, the size of the convolutional kernels is limited to 1×1 to model patterns of correlations in small local regions.

The main differences between PNet (VNet) and XuNet can be summarized as follows:

(1) Group 1 consists of two concatenated convolutional layers (1F) and (1T). The 1F layer consists of four 5×5 high pass kernels whose parameters are fixed during training. The layer 1T contains eight 5×5 kernels whose parameters are learned during training. The fixed kernels increase the SNR between the cover image and the stego signal. They also act as regularizers to narrow down the feasible feature space and thus help facilitate the convergence of the network.
(2) There is no pooling in Groups 1 and 2 because it would lead to phase loss and mixing.

(3) We insert a PhaseSplit module between Groups 2 and 3, that splits Group 2's outputs $16\times(512\times512)$ by their JPEG phase into 64 groups of $16\times(64\times64)$. In other words, each 512×512 feature map will be split into 64 64×64 maps, one map per JPEG phase.
(4) The last three layers are similar to XuNet but each of the 64 JPEG feature groups consisting of 16×(64×64) feature maps are first convolved with $32\times(1\times1)$ kernels to get $32\times(32\times32)$ feature maps with ReLU and average pooling by 2. In the next layer, convolution with $64\times(1\times1)$ kernels leads to $64\times(16\times16)$ feature maps with ReLU and average pooling by 2. And, in the last convolutional layer, convolutions with $128\times(1\times1)$ kernels are applied to obtain a $128\times(1\times1)$ feature map with ReLU and average pooling by 16. In a PNet, each phase group is processed independently, which means that in the end there are 64×128 outputs fully connected to the "classifier" part of the network.

Note that the 64 parallel channels do not need to be implemented as a directed acyclic graph. They can be implemented by chaining layers by using the convolutional filter group options supported by most CNN packages. To be more specific, the convolutional module computes the convolution of the input map $\mathbf{x}$ with a bank of D" multi-dimensional filters $\mathbf{f}$ to get the output $\mathbf{y}$. Here, $\mathbf{x}\in\mathbb{R}^{H\times W\times D}$, $\mathbf{f}\in\mathbb{R}^{D''\times(H'\times W'\times D)}$, $\mathbf{y}\in\mathbb{R}^{(H''\times W''\times D'')}$, where the data $\mathbf{x}$ and $\mathbf{y}$ have a spatial structure (height×width×number of feature maps), and the filter $\mathbf{f}$ has the structure output feature map×(height×width×number of input feature maps). The filter group option is the flexibility to group input feature maps $\mathbf{x}$ and apply different subsets of filters to each group. To use this feature, one specifies as input a bank of D" filters $\mathbf{f}\in\mathbb{R}^{D''\times(H'\times W'\times D')}$ such that D' divides the number of input dimensions D. These are treated as $g=D/D'$ filter groups; the first group is applied to dimensions $d=1,\ldots,D'$ of the input $\mathbf{x}$; the second group is applied to dimensions $d=D'+1,\ldots,2D'$ of the input $\mathbf{x}$, and so on.

3.3 PhaseSplit module

The JPEG phase-split module is illustrated in Fig. 2. Each feature map $\mathbf{x}$ is subsampled on 64 sublattices $\mathcal{L}_{ij}$ (1) obtaining thus 64 feature submaps (for each sublattice $\mathcal{L}_{ij}$) from each feature map $\mathbf{x}$.

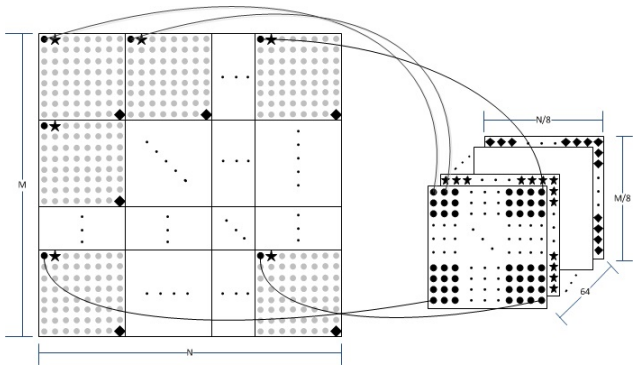


Figure 2: PhaseSplit module.

3.4 Refined batch normalization

Batch normalization is an important design element for training the proposed CNNs. Its allows us to use a larger learning rate and thus speed up the learning, making the training process less dependent on proper initialization. It also improves the detection accuracy.

The standard way to learn the BN's per-channel mean and standard deviation is via moving average during training. However, our architecture has too many BN parameters. For example, the PNet has 8 + 16 + 2048 + 4096 + 8192 = 14,360 pairs of mean and standard deviation. Small errors propagate to the end and diminish the performance. This "moment drift" is especially undesirable in steganalysis because the stego signal is very week. The consequence of this are large fluctuations in the validation error during training. To alleviate this problem, we refined the BN moment learning in the following manner:

(1) Train the network as usual, without paying attention to the BN moment learning rate.
(2) Every 8–10 epochs, freeze all net parameters and put the net into the evaluation mode. Feed it with 2000/ 3000 mini-batches from the randomized training set and calculate and record the BN moments (mean and standard deviation) for all these mini-batches.
(3) Set the BN moments to the median values across the computed moments, validate, and test.

To further improve the accuracy, the batch size used in this refinement step can be chosen to be a very large number compared with the one used during training since there is no need for backward propagation of the gradients during this step.

3.5 Fixed linear filters

The four 5×5 high pass kernels in the convolutional layer 1F are shown in Fig. 3. The first filter, F_{KV}, is 'SQUARE5x5' from the Spatial Rich Model (SRM) [6], also used in one of the first CNN steganalysis detectors [18] and in XuNet. Due to its sign-changing symmetric checkerboard pattern, this filter suppresses correlated components (image content) while largely preserving the high-frequency stego signal. The second filter F_{P} is a point high-pass filter chosen to complement F_{KV}. It acts like a catalyst (activator in chemical reactions) and allows the network to learn the kernels in the second layer that are better at extracting the stego noise introduced by JPEG steganography as will be seen below. The F_{H} and F_{V} are two second-order horizontal and vertical Gabor filters (2), with $\varphi=\pi/2,\sigma=1,\theta=0$ and $\theta=\pi/2$, respectively. These two filters are added because F_{KV} and F_{P} are not directional.

Fig. 4 shows the effect of different combinations of fixed kernels on the detection error for our standard developer setup (Sec. 3.1). With only the F_{KV} kernel, the detection error rate saturates at about 18–19%. However, after adding the catalyst high-pass filter F_{P}, the detection error rate quickly drops to 8.6%. With the addition of F_{H} and F_{V}, the error rate further decreases to 7.9%.

Leveraging the fact that convolution is associative and distributive, the combined action of the convolutional layers 1F and 1T is equivalent to convolutions with eight 9×9 kernels. The equivalent kernels of the CNN trained with only F_{KV} and with both F_{KV} and F_{P} are illustrated in Fig. 5, which sheds light on the role of the point filter F_{P}. With only F_{KV}, the trained 9×9 kernels in Group 1 still

$$F_{\mathrm{KV}} = \begin{bmatrix} -.0833 & +.1667 & -.1667 & +.1667 & -.0833 \\ +.1667 & -.5000 & +.6667 & -.5000 & +.1667 \\ -.1667 & +.6667 & -1.0000 & +.6667 & -.1667 \\ +.1667 & -.5000 & +.6667 & -.5000 & +.1667 \\ -.0833 & +.1667 & -.1667 & +.1667 & -.0833 \end{bmatrix}$$

(a) KV filter.

$$F_{\mathrm{P}} = \begin{bmatrix} 0 & 0 & +.0199 & 0 & 0 \\ 0 & +.0897 & +.1395 & +.0897 & 0 \\ -.0199 & +.1395 & -1.0000 & +.1395 & +.0199 \\ 0 & +.0897 & +.1395 & +.0897 & 0 \\ 0 & 0 & +.0199 & 0 & 0 \end{bmatrix}$$

(b) Point high-pass filter.

$$F_{\mathrm{H}} = \begin{bmatrix} +.0562 & -.1354 & 0 & +.1354 & -.0562 \\ +.0818 & -.1970 & 0 & +.1970 & -.0818 \\ +.0926 & -.2233 & 0 & +.2233 & -.0926 \\ +.0818 & -.1970 & 0 & +.1970 & -.0818 \\ +.0562 & -.1354 & 0 & +.1354 & -.0562 \end{bmatrix}$$

(c) Horizontal 2D Gabor filter.

$$F_{\mathrm{V}} = \begin{bmatrix} -.0562 & -.0818 & -.0926 & -.0818 & -.0562 \\ +.1354 & +.1970 & +.2233 & +.1970 & +.1354 \\ 0 & 0 & 0 & 0 & 0 \\ -.1354 & -.1970 & -.2233 & -.1970 & -.1354 \\ +.0562 & +.0818 & +.0926 & +.0818 & +.0562 \end{bmatrix}$$

(d) Vertical 2D Gabor filter.

Figure 3: Four fixed 5×5 kernels used in the first convolutional layer (1F).

adhere to the checkerboard pattern similar to F_{KV} (c.f. a similar observation made in [21]). However, with the introduction of F_{P}, they exhibit a correlated structure more suitable for detection of stego signal spanned by discrete cosines. To put this another way, while the F_{KV} filter works well for detection of high-frequency stego signals introduced by typical spatial-domain embedding schemes, the impact of embedding in the JPEG domain examined in the spatial domain is very different because the difference between cover and stego images is due to changes to quantized DCT coefficients. And because most DCT coefficients that are modified are in the low to medium frequency band, the nature of the stego signal is different. High-pass filters, such as F_{KV}, suppress together with the content also the stego signal. In fact, this is most likely why Gabor filters work better for steganalysis of JPEG files than SRM high-pass filters.

As our last note in this section and to complete the investigation of the most appropriate structure of the 1F layer, we tested whether another improvement can be obtained by including a larger set of Gabor kernels in this layer. Specifically, we formed the 1F layer with eight 8×8 filters obtained from Eq. (2) with parameters $\sigma = 1$, $\varphi = [0, \pi/2]$, and $\theta = [0, \pi/4, \pi/2, 3\pi/4]$. In our standard developer setup, a PNet with these fixed filters, however, lagged 2% behind the PNet with the above mentioned four fixed kernels.

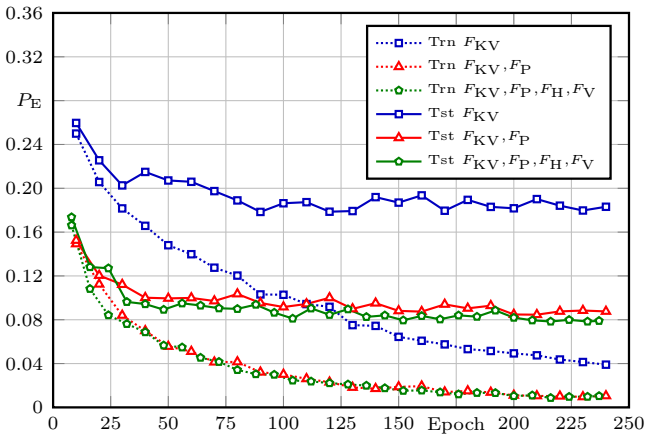


Figure 4: Detection error with different filter combinations under standard developer setup.

Table 1: Detection error (in %) of three PNet configurations under standard developer setup.

	$C_{1\mathrm{F}}4 \times C_{1\mathrm{T}}8 \times C_2 16$	$C_{1\mathrm{F}}4 \times C_{1\mathrm{T}}12 \times C_2 16$	$C_{1\mathrm{F}}4 \times C_{1\mathrm{T}}12 \times C_2 24$
BOSSbase (8,000)	7.91±0.03	8.10±0.10	8.08±0.11
BOWS2 (20,000)	9.15±0.09	8.91±0.07	9.09±0.18

3.6 Configuration before phase split

Fixing the parameters of the fixed filter (1F) and the portion of the net after the PhaseSplit module, we investigate if further performance gain is possible by deepening the architecture, inserting more convolutional layers, and by widening the net. We abbreviate the net architecture in the first two groups by the number of convolutional feature maps. For example, the configuration in Fig. 1 left is named $C_{1\mathrm{F}}4 \times C_{1\mathrm{T}}8 \times C_2 16$. The following alternatives were studied:

(1) Increased number of kernels (1T) in Group 1 from $8 \times (5 \times 5)$ to $12 \times (5 \times 5)$, abbreviated as $C_{1\mathrm{F}}4 \times C_{1\mathrm{T}}12 \times C_2 16$.
(2) Increased the number of kernels (1T) in Group 1 from $8 \times (5 \times 5)$ to $12 \times (5 \times 5)$ and the number of kernels in Group 2 from $16 \times (5 \times 5)$ to $24 \times (5 \times 5)$, abbreviated $C_{1\mathrm{F}}4 \times C_{1\mathrm{T}}12 \times C_2 24$.

The impact of these modifications on the PNet was evaluated on the standard developer setup (Sec. 3.1) with the results shown in Table 1. In summary, we did not observe any statistically significant gain by enlarging the configuration of the PNet.

3.7 PNet vs. VNet

In this section, we compare the performance of PNet and VNet from Fig. 1. For a fair comparison, both detectors were trained with identical training and testing settings with the standard developer setup. The training and testing detection errors are shown in Fig. 6 and in Table 3. The ensembles of five nets were obtained by randomly splitting the training set into five disjoint subsets, training on four of them and using the fifth for validation as described in [29]. In

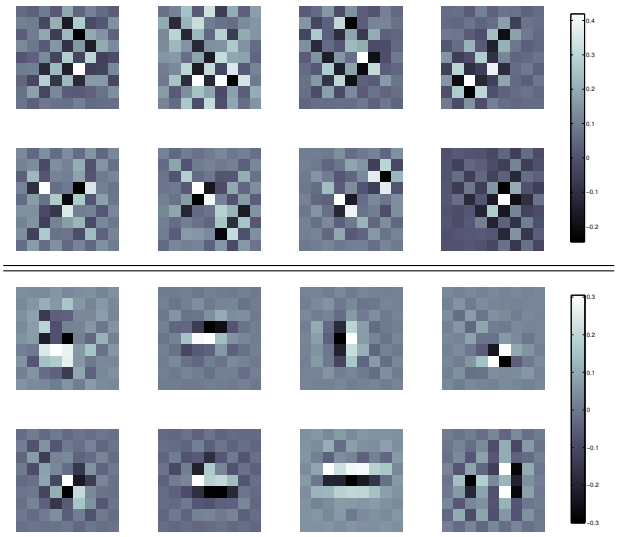


Figure 5: Convolutions of F_{KV} with the kernels from the second convolutional layer (1T) equivalent to eight 9×9 kernels of the convolutional module in Group 1. Top: Kernels learned with only F_{KV}. Bottom: learned with both F_{KV} and F_{P}.

contrast to [29], in our ensemble we applied the majority voting rule on the decisions returned by the individual five nets.

The individually trained PNet is 1.1% better than the VNet when tested on both BOSSbase (5,000 images) and BOWS2 (20,000 images). The ensemble of five PNets improves upon the individual PNet by 1.04%, from 7.60% to 6.56%. On the other hand, since VNets are more over-trained (see Fig. 6), the ensemble of five VNets produces a larger boost (1.65%), decreasing the detection error of the ensemble of VNets from 8.70% to 7.05%. Therefore, the ensemble of five PNets is only 0.49% better than VNets. Similarly, when testing on BOWS2, the average testing detection error for PNets and VNets are 9.01% and 10.11%, respectively. The ensemble of five PNets is 7.66%, while the ensemble of VNets is 8.07%.

In summary, PNet gives a slightly better performance than VNet but is also more computationally demanding and takes longer to train (about two times longer). Table 2 contrasts the complexity of the two explored architectures. In the end, the specific demands of the application should determine which detector type is more suitable.

The experiments in this section also indicate that the cover source mismatch between BOSSbase and BOWS2 does not degrade the performance of either net architecture by any significant amount. In contrast, the cover source mismatch in the spatial domain had a bigger negative impact on XuNet [30]. This is most likely due to the fact that JPEG compression has a tendency to equalize differences between sources.

Table 2: Net structure complexity of PNet and VNet under standard developer setup. The speed is evaluated on two GTX 1080 GPU cards using MatConvNet and batch size 40.

	Data Memory	Parameters Memory	Train Speed	Test Speed
PNet	11 GB	7.7e+05	25.4 images/sec	100.4 images/sec
VNet	4 GB	3.0e+05	57.1 images/sec	218.1 images/sec

Table 3: Detection error (in %) of PNet and VNet and their bagging ensembles under standard developer setup.

	Individual PNets	Ensemble PNets	Individual VNets	Ensemble VNets
BOSSbase	7.60±0.24	6.56±0.10	8.70±0.28	7.05%±0.09
BOWS2	9.01±0.24	7.66±0.08	10.11±0.41	8.07%±0.10

4 EXPERIMENTS

In this section, we report the detection accuracy of the proposed JPEG-phase-aware CNNs and interpret the results.

4.1 Dataset and software platforms

Two JPEG-domain content-adaptive steganographic algorithms were tested: J-UNIWARD and UED-JC with a range of embedding rates. The results are contrasted with what can be achieved using conventional feature-based steganalysis with SCA-GFR features [5] with the FLD-ensemble [16]. All experiments using the CNNs were performed with a modified MatConvNet toolbox [27]. All tests are done using GTX 1080 and Titian X GPU cards.

As in the standard developer setup, we used BOSSbase 1.01 for training all detectors. The testing was done on the unused part of BOSSbase and in some experiments on BOWS2 to see the robustness of the detectors to cover source mismatch.

4.2 Training, validation, and testing

For each experiment reported in Table 4, BOSSbase was randomly split into 75/25 ratio. To build the ensembles, the training set was furthermore randomly split into five equally sizes folds, using four of them for training and the fifth fold for validation (bagging). Thus, each CNN detector was trained on 6,000 cover-stego pairs and validated on 1,500 pairs.

The testing is carried out on the remaining 2,500 cover-stego pairs from BOSSbase images and also on the entire BOWS2 database. The detection error and the statistical spread of each individual network is computed as the mean and standard deviation of error rates from the last five evaluation steps of each network during training using refined batch normalization (Section 3.4). The corresponding values for ensemble networks are computed over error rates from the last five steps of five different splits of the training set (a total of 25 values).

4.3 Hyper-parameters

Mini-batch stochastic gradient descent was used to train all CNNs. The momentum was fixed to 0.9, the learning rate for all parameters was initialized to 0.001 and scheduled to decrease 25% every 20 epochs ($20 \times 300 = 6,000$ iterations), up to 240 epochs. A mini-batch

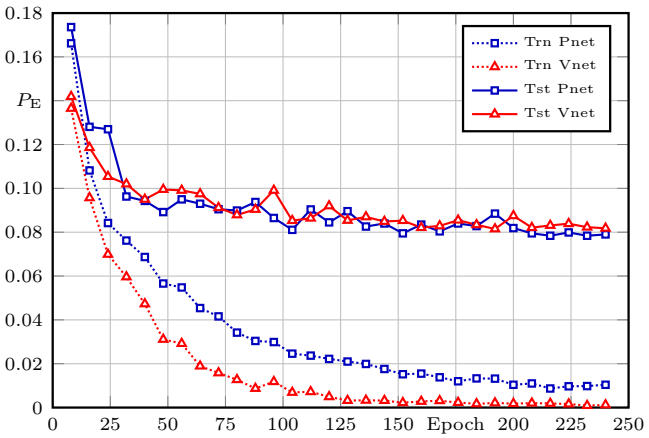


Figure 6: Detection error of PNets and VNets as a function of training epochs under standard developer setup.

of 40 images (20 cover/stego pairs) was input for each iteration.[1] The training database is also randomly shuffled at the beginning of each epoch to help batch normalization and give the CNN detector a better ability to generalize. The parameters of convolution kernels were initialized from a zero-mean Gaussian distribution with standard deviation 0.01. As in XuNet [30], the bias learning in all convolutional layers, except for the last fully connected layer whose bias learning rate is set to be twice of the learning rate, were fixed to zero to make the feature maps sign symmetric. The parameters in the last fully-connected (FC) layers were initialized using Xavier initialization [8]. The weight decay (L2 regularization) was not enabled, except for the FC layer, which was set to 0.01.

For BN layers, the learning rate is the same as the global learning rate of the network with all scale and shift parameters initialized with 1 and 0, respectively. As explained in Sec. 3.4, the per-channel mean and standard deviation in BN were implemented using the moment refinement performed every 8 (or 4) epochs during training, which gives the net 2000 random mini-batches of size 48 from the training set to compute the refined moments.

4.4 Curriculum learning for steganalyzing different payloads

Due to the PhaseSplit module and the catalyst filter kernel, both PNets and VNets converge relatively quickly even for small payloads, such as 0.1 bpnzac, and high JPEG quality. The training can be further sped up by adopting the transfer learning strategy [19, 31] widely used in CNNs. In particular, for each steganographic scheme and JPEG quality factor, we first train a CNN on a training set embedded with 0.4 bpnzac payload (from randomly initialized weights). To train a detector for the other payloads (0.1–0.3, and 0.5), we initialized with the model trained on 0.4 bpnzac and then fine-tuned this model on the same training set embedded with the corresponding payload. Curriculum learning enables us to decrease the number of training epochs by a factor of two.

Our experiments consistently show that the use of a pre-trained CNN outperforms, or in the worse case, performs as well as a CNN trained from randomly initialized weights. In addition, it is also possible to train CNNs with the pre-trained CNNs of other steganographic schemes, for example, to train UED-JC CNN detector, we can use a CNN pre-trained on J-UNIWARD.

[1] As shown in Table 2, the memory requirement for PNet with batch size 40 is more than 11 GB, whichexceeds the GPU memory of any single common GPU card. This problem is solved by employing two GPUs for training. More specifically, the data (40 images) are further divided in two sub-batches of 20 images and distributed among two GPUs with some communication overhead.

Table 4: Detection error comparison between individual (Ind.) and ensemble of (Ens.) PNet(s) and VNet(s) with SCA-GFR features on J-UNIWARD and UED-JC.

(a) Detection error (in %) for J-UNIWARD

QF	Detector	0.1	0.2	0.3	0.4	0.5
75	SCA GFR	35.54±0.54	22.47±0.38	13.44±0.28	7.53±0.21	4.15±0.13
	Ind. PNet	37.26±0.31	23.50±0.34	13.80±0.41	7.60±0.24	4.13±0.27
	Ens. PNets	35.75±0.25	21.26±0.32	12.28±0.13	6.56±0.10	3.36±0.05
	Ind. VNet	37.59±0.45	24.57±0.54	15.05±0.41	8.70±0.28	4.74±0.24
	Ens. VNets	36.15±0.16	22.40±0.17	13.32±0.08	7.05±0.09	3.74±0.14
95	SCA GFR	46.03±0.35	40.07±0.48	32.92±0.31	25.54±0.63	19.35±0.54
	Ind. PNet	46.76±0.22	41.52±0.50	34.74±0.84	28.01±1.14	20.42±1.15
	Ens. PNets	45.89±0.29	39.89±0.17	31.91±0.11	25.36±0.23	17.49±0.15
	Ind. VNet	47.43±0.51	43.65±1.49	34.76±0.41	28.02±0.40	21.31±0.25
	Ens. VNets	47.07±0.22	42.73±0.58	33.28±0.12	25.93±0.21	19.67±0.15

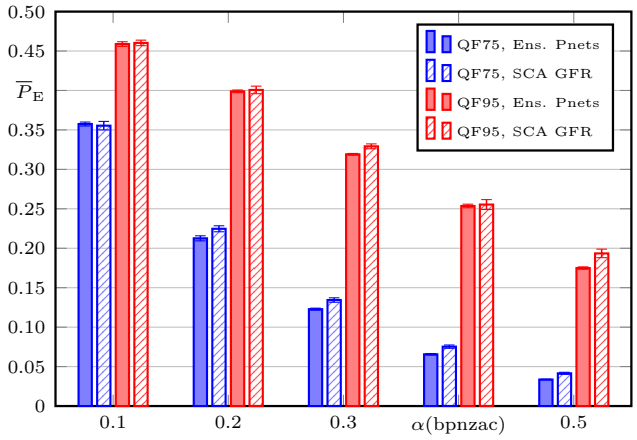
(b) Detection error (in %) for UED-JC

QF	Detector	0.1	0.2	0.3	0.4	0.5
75	SCA GFR	22.54±0.67	11.56±0.32	6.35±0.20	3.46±0.14	1.74±0.19
	Ind. PNet	20.11±0.38	9.55±0.21	4.56±0.23	2.81±0.17	1.48±0.17
	Ens. PNets	17.77±0.31	8.52±0.11	3.90±0.05	2.34±0.43	1.33±0.02
	Ind. VNet	21.62±0.44	10.07±0.46	5.48±0.10	3.07±0.15	1.63±0.18
	Ens. VNets	18.97±0.16	8.04±0.12	4.07±0.04	2.32±0.03	1.20±0.02
95	SCA GFR	39.20±0.51	30.48±0.62	22.55±0.61	15.92±0.48	10.52±0.45
	Ind. PNet	39.27±0.53	29.93±0.59	21.36±1.10	15.00±0.91	9.51±1.04
	Ens. PNets	37.18±0.36	27.21±0.37	18.46±0.39	12.27±0.34	7.31±0.13
	Ind. VNet	42.00±1.11	31.52±1.41	22.39±0.62	15.75±0.51	9.98±0.45
	Ens. VNets	40.63±0.13	28.55±0.19	19.46±0.23	13.18±0.13	7.95±0.04

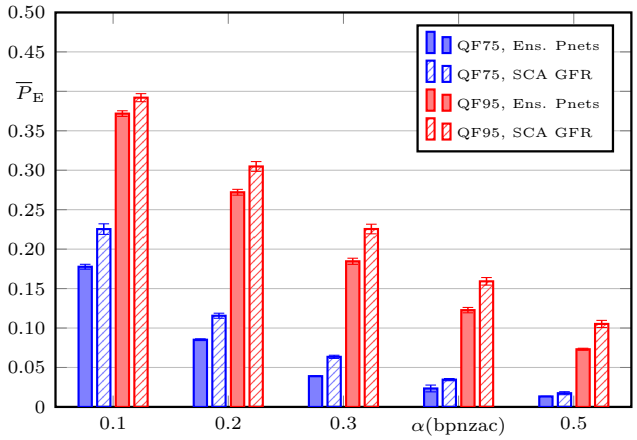
4.5 Analysis

Below, we summarize lessons learned and the main results of our experiments.

- Batch normalization with proper moment refinement plays an important role in training JPEG-phase-aware CNNs because it normalizes each individual phase at different stages. Because the training batches are small, the moments computed over training and testing sets exhibit large fluctuations, makes it unclear when the network converged. The refined BN smooths the training and validation performance curves, helps find better convergence points, and also improves the model accuracy by 3–7% over models trained using moving average.
- While the individual phase-aware nets are slightly inferior to the SCA GFR for J-UNIWARD, an ensemble of five individually trained nets improves upon SCA GFR by 1%.



(a) J-UNIWARD



(b) UED-JC

Figure 7: Performance comparison between the ensemble of five PNets and SCA GFR for two embedding schemes.

- For UED-JC, the proposed CNN detectors clearly outperform conventional steganalysis. In general, each trained individual net is better than the SCA GFR with the FLD-ensemble. The ensemble of five PNets (or VNets) brings further improvement.

4.6 On the net transfer ability

In this section, we study the ability of the detector trained on one embedding algorithm to detect a different embedding algorithm to see how well it transfers to a different stego source. In particular, we trained a PNet on examples of cover and stego images embedded with J-UNIWARD and then tested it on UED-JC stego images, all stego images at 0.4 bpnzac (see Table 5). Although the test error rate 3.79% is higher when compared to the detector trained for UED-JC (2.3–2.4%), the detection accuracy drop is rather small. On the other hand, a PNet trained on UED-JC stego images performs rather poorly on J-UNIWARD images; the error rate is around

Table 5: Detection error (in %) to evaluate the transfer ability of three detectors when training on J-UNIWARD images and testing on UED-JC images and vice versa. Standard developer setup.

Detector	TRN Images	J-UNI TST Images	UED TST Images
SCA GFR	J-UNI	7.53±0.21	6.07±0.21
Ind. PNet	J-UNI	7.60±0.24	4.66±0.28
Ens. PNets	J-UNI	6.56±0.10	3.79±0.10
SCA GFR	UED	12.19±0.70	3.46±0.14
Ind. PNet	UED	27.50±1.51	2.81±0.17
Ens. PNets	UED	27.25±0.40	2.34±0.43

27.50%. This indicates that UED-JC introduces artifacts that the PNet discovers and that J-UNIWARD avoids. We hypothesize that a portion of the weights in the model trained on UED-JC become noticeably larger than the rest as the network focuses on a portion of the "features" that are strongly related to the artifacts created by UED embedding. The network thus basically "overtrains to UED" and does not generalize well to J-UNIWARD. This could likely be alleviated by introducing dropout.

For comparison, in the table we also report the performance of an FLD-ensemble trained on SCA GFR features. While this detector does not generalize as well as the PNet to UED test images when trained on J-UNIWARD, the loss observed when training on UED and testing on J-UNIWARD images is much smaller (12.19% vs. 27.50 for the PNet).

5 CONCLUSION AND FUTURE WORK

Convolutional neural networks (CNNs) only recently e-merged in the field of steganalysis as a competitive alternative to the "classical" approach based on training a classifier on features extracted from examples of cover and stego images. Since the moment they were introduced by Qian et al. in 2015 [18], they were touted as "fully automatic" tools in the sense of not requiring hand crafted features. The authors of this article feel that we are still rather far away from this ultimate goal of fully automatizing the process of developing steganalysis detectors. Human insight currently remains an indispensable element in designing network architectures and insight painstakingly gathered from classical approach to steganalysis is still extremely valuable. This paper only seems to confirm this thesis.

Inspired by the success of features aware of the JPEG phase, we introduce phase awareness into the architecture of a CNN to see whether current state-of-the-art steganalysis tools for detection of modern JPEG steganography can be improved. To this end, we started with a structural CNN design proposed by Xu et al. [29, 30] for spatial domain steganalysis and adjusted it for steganalysis in the JPEG domain in two different ways. The result are two network architectures, the PNet and VNet. While PNet gives slightly better results than VNet, the latter is faster to train and test and enjoys smaller memory requirements. We leave the choice of the detector in practice to the specifics required by the application.

While our investigation focused on phase awareness, along the way we discovered another interesting phenomenon,

which we termed "catalytic kernel." Stego signal is a high-frequency noise modulated by content. Thus, to force the network to pay attention to this signal (and to converge), images need to be pre-filtered to incease the SNR between the signal of interest (the weak stego noise) and the noise – the image content. Fixed high-pass filters are usually used for this job in current network designs. While sign-changing high-pass filters, such as the ubiquitous "KV filter" work well for steganalysis of spatial-domain steganography, the stego signal introduced in the spatial domain by modifying quantized DCT coefficients is no longer high frequency in nature. After all, it is spanned by quantized discrete cosines and thus "inhabits" medium and low spatial frequencies as well. Pre-filtering images with filters, such as the KV, thus inevitably and undesirably suppresses the stego signal as well. To help the network find more suitable filters, we supply in the fixed layer another "point filter" that seems to play the role of a catalyst (regularizer) that allows the filters in the first convolutional layer to accept shapes more suitable for detecting stego noise caused by JPEG steganography. Just augmenting the KV filter with this point catalytic filter brought the detection performance of our networks from mediocre to highly competitive. We hypothesize that a similar effect may be achieved with the constrained convolutional layer recently proposed by Bayar et al. [3]. A more detailed investigation of this is postponed to our future effort.

In summary, we showed that JPEG phase awareness can be built into CNN architectures and that it indeed improves detection accuracy over classical steganalysis. The observed gain was significantly larger for UED-JC than for J-UNI-WARD. We also investigated the ability of the trained networks to generalize to previously unseen cover and stego sources.

All code used to produce the results in this paper, including the network configuration files, the PhaseSplit Layer, and the refined Batch Normalization layer is available from http://dde.binghamton.edu/download/ for download.

ACKNOWLEDGMENTS

The work on this paper was supported by Air Force Office of Scientific Research under the research grant number FA9950-12-1-0124. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of AFOSR or the U.S. Government. The authors would like to thank anonymous reviewers for their insightful comments.

REFERENCES

[1] P. Bas, T. Filler, and T. Pevný. 2011. Break Our Steganographic System – the Ins and Outs of Organizing BOSS. In *Information Hiding, 13th International Conference (*Lecture Notes in Computer Science), T. Filler, T. Pevný, A. Ker, and S. Craver (Eds.), Vol. 6958. Springer Berlin Heidelberg, Prague, Czech Republic, 59–70.

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

[3] B. Belhassen and M. C. Stamm. 2016. A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. In *The 4th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '16)*, F. Perez-Gonzales, F. Cayre, and P. Bas (Eds.). Vigo, Spain, 5–10.

[4] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich. 2014. Selection-Channel-Aware Rich Model for Steganalysis of Digital Images. In *IEEE International Workshop on Information Forensics and Security*. Atlanta, GA.

[5] T. D. Denemark, M. Boroumand, and J. Fridrich. 2016. Steganalysis Features for Content-Adaptive JPEG Steganography. *IEEE Transactions on Information Forensics and Security* 11, 8 (August 2016), 1736–1746. https://doi.org/10.1109/TIFS.2016.2555281

[6] J. Fridrich and J. Kodovský. 2011. Rich Models for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics and Security* 7, 3 (June 2011), 868–882.

[7] J. Fridrich, T. Pevný, and J. Kodovský. 2007. 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, 3–14.

[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS '10)*. Sardinia, Italy, 249–256.

[9] L. Guo, J. Ni, and Y.-Q. Shi. 2012. An Efficient JPEG Steganographic Scheme Using Uniform Embedding. In *Fourth IEEE International Workshop on Information Forensics and Security*. Tenerife, Spain.

[10] L. Guo, J. Ni, and Y. Q. Shi. 2014. Uniform Embedding for Efficient JPEG Steganography. *IEEE Transactions on Information Forensics and Security* 9, 5 (May 2014), 814–825. https://doi.org/10.1109/TIFS.2014.2312817

[11] S. Hetzl and P. Mutzel. 2005. A Graph-Theoretic Approach to Steganography. In *Communications and Multimedia Security, 9th IFIP TC-6 TC-11 International Conference, CMS 2005 (*Lecture Notes in Computer Science), J. Dittmann, S. Katzenbeisser, and A. Uhl (Eds.), Vol. 3677. Salzburg, Austria, 119–128.

[12] V. Holub and J. Fridrich. 2015. Low-Complexity Features for JPEG Steganalysis Using Undecimated DCT. *IEEE Transactions on Information Forensics and Security* 10, 2 (February 2015), 219–228.

[13] V. Holub and J. Fridrich. 2015. 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, 0T 1–11.

[14] V. Holub, J. Fridrich, and T. Denemark. 2014. 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* 2014:1 (2014).

[15] J. Kodovský and J. Fridrich. 2012. 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, 0A 1–13.

[16] J. Kodovský, J. Fridrich, and V. Holub. 2012. Ensemble Classifiers for Steganalysis of Digital Media. *IEEE Transactions on Information Forensics and Security* 7, 2 (2012), 432–444.

[17] N. Provos. 2001. Defending Against Statistical Steganalysis. In *10th USENIX Security Symposium*. Washington, DC, 323–335.

[18] Y. Qian, J. Dong, W. Wang, and T. Tan. 2015. Deep learning for steganalysis via convolutional neural networks. In *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, A. Alattar and N. D. Memon (Eds.), Vol. 9409. San Francisco, CA, 0J 1–10.

[19] Y. Qian, J. Dong, W. Wang, and T. Tan. 2016. Learning and transferring representations for image steganalysis using convolutional neural network. In *IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, 2752–2756.

[20] P. Sallee. 2005. Model-Based Methods for Steganography and Steganalysis. *International Journal of Image Graphics* 5, 1 (2005), 167–190.

[21] V. Sedighi and J. Fridrich. 2017. 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.). Burlingame, CA.

[22] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang. 2015. Steganalysis of Adaptive JPEG Steganography Using 2D Gabor Filters. In *The 3rd ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '15)*, A. Alattar, J. Fridrich, N. Smith, and P. Comesana Alfaro (Eds.). Portland, OR.

[23] S. Tan and B. Li. 2014. Stacked convolutional auto-encoders for steganalysis of digital images. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. 1–4. https://doi.org/10.1109/APSIPA.2014.7041565

[24] T. Thai, R. Cogranne, and F. Retraint. 2014. Statistical Model of Quantized DCT Coefficients: Application in the Steganalysis of Jsteg Algorithm. *IEEE Transactions on Image Processing* 23, 5 (May 2014), 1–14.

[25] T. H. Thai, R. Cogranne, and F. Retraint. 2014. Optimal Detection of OutGuess using an Accurate Model of DCT Coefficients. In *Sixth IEEE International Workshop on Information Forensics and Security*. Atlanta, GA.

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

[27] A. Vedaldi and K. Lenc. 2014. MatConvNet - Convolutional Neural Networks for MATLAB. *CoRR* abs/1412.4564 (2014). http://arxiv.org/abs/1412.4564

[28] A. Westfeld. 2001. High Capacity Despite Better Steganalysis (F5 – A Steganographic Algorithm). In *Information Hiding, 4th International Workshop (*Lecture

Notes in Computer Science), I. S. Moskowitz (Ed.), Vol. 2137. Springer-Verlag, New York, Pittsburgh, PA, 289–302.

[29] G. Xu, H.-Z. Wu, and Y. Q. Shi. 2016. Ensemble of CNNs for Steganalysis: An Empirical Study. In *The 4th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '16)*, F. Perez-Gonzales, F. Cayre, and P. Bas (Eds.). Vigo, Spain, 5–10.

[30] G. Xu, H. Z. Wu, and Y. Q. Shi. 2016. Structural Design of Convolutional Neural Networks for Steganalysis. *IEEE Signal Processing Letters* 23, 5 (May 2016), 708–712. https://doi.org/10.1109/LSP.2016.2548421

[31] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. 2014. How Transferable Are Features in Deep Neural Networks?. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*. Cambridge, MA, 3320–3328.

[32] T. Zhang and X. Ping. 2003. A Fast and Effective Steganalytic Technique Against Jsteg-like Algorithms. In *Proceedings of the ACM Symposium on Applied Computing*. Melbourne, FL, 307–311.