Yedroudj-Net: An Efficient CNN for Spatial Steganalysis
Mehdi Yedroudj, Frédéric Comby, Marc Chaumont

To cite this version:

HAL Id: lirmm-01717550
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01717550
Submitted on 26 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
YEDROUDJ-NET: AN EFFICIENT CNN FOR SPATIAL STEGANALYSIS

Mehdi YEDROUDJ (1), Frédéric COMBY (1), Marc CHAUMONT (1,2)

(1) Montpellier University, LIRMM (UMR5506) / CNRS, FRANCE, (2) Nîmes University, FRANCE.

ABSTRACT

For about 10 years, detecting the presence of a secret message hidden in an image was performed with an Ensemble Classifier trained with Rich features. In recent years, studies such as Xu et al. have indicated that well-designed convolutional Neural Networks (CNN) can achieve comparable performance to the two-step machine learning approaches.

In this paper, we propose a CNN that outperforms the state-of-the-art in terms of error probability. The proposition is in the continuity of what has been recently proposed and it is a clever fusion of important bricks used in various papers. Among the essential parts of the CNN, one can cite the use of a pre-processing filter bank and a Truncation activation function, five convolutional layers with a Batch Normalization associated with a Scale Layer, as well as the use of a sufficiently sized fully connected section. An augmented database has also been used to improve the training of the CNN.

Our CNN was experimentally evaluated against S-UNIWARD and WOW embedding algorithms and its performances were compared with those of three other methods: an Ensemble Classifier plus a Rich Model, and two other CNN steganalyzers.

Index Terms— Steganalysis, Deep Learning, Convolutional Neural Network.

1. INTRODUCTION

The first attempt to use Deep Learning methods for steganalysis dates back to 2014 [1] with auto-encoders. One year later Qian et al. [2] and Pibre et al. [3] proposed to use Convolutional Neural Networks. In 2016, the first results, close to those of the state-of-the-art, were obtained with an ensemble of CNNs [4]. The Xu-Net [5] works. In 2016, the first results, close to those of the state-of-the-art, is used as base learner et al. [2] and Pibre [3] proposed to use Convolutional Neural Networks. In recent years, studies such as Xu et al. [12] and Pibre [13] have been proposed in 2017, this time for JPEG steganalysis. In [6], authors proposed a pre-processing inspired by the Rich Models, and the use of a big learning database. The results were close to those of the state-of-the-art. In [7], the network is built with a phase-split inspired by the JPEG compression process. The results are close to those of the state-of-the-art. In [8], authors proposed a pre-processing inspired by the Rich Models and the use of a big learning database. The results were close to those of the state-of-the-art. In [7], we proposed to use an Ensemble Classifier that transforms the input image into a feature vector and a classification module, consisting of three fully connected layers followed by a softmax. The network produces a probability distribution over the two class labels.

The pre-processing block filters the input covert/stego image with a predefined high-pass filter in order to extract the noise component residuals. The pre-processed image then feeds the network. Previous studies [2, 3] observed that without this preliminary high-pass filter the CNN converges more slowly. This pre-processing largely suppresses the image content, narrows the dynamic range, and thus increases the signal-to-noise ratio between the weak stego signal (if present) and the image signal. As a result, the CNN can learn on a more compact and robust signal.

2. YEDROUDJ-NET

Fig. 1 illustrates the overall architecture of our CNN. The network is composed of a pre-processing block, five convolutional blocks, and a fully connected block made of three fully connected layers followed by a softmax. The network produces a probability distribution over the two class labels.

The pre-processing block filters the input covert/stego image with a predefined high-pass filter in order to extract the noise component residuals. The pre-processed image then feeds the network. Previous studies [2, 3] observed that without this preliminary high-pass filter the CNN converges more slowly. This pre-processing largely suppresses the image content, narrows the dynamic range, and thus increases the signal-to-noise ratio between the weak stego signal (if present) and the image signal. As a result, the CNN can learn on a more compact and robust signal.

Inspired by the benefit of diversity [12], and similarly to [17], we use the 30-basic high-pass filters from SRM [12], instead of using only one filter such as [2, 3, 5], in order to pre-process the input image. Note that the filters kernel values of the pre-processing block, i.e. the weights, are not optimized/learned during the training. This pre-processing has been integrated into a lazy fashion, directly into the CNN, such that the size of all kernels (weighting matrix) are set to 5×5. Their central part is initialized with the weights of the SRM kernels and the remaining elements are padded to zero. No normalization of the kernels’ values is performed.

The rest of our CNN can be divided into a convolutional module, dedicated to features representation, that transforms the input image into a feature vector, and a classification module, consisting...
of three fully-connected layers and a softmax layer, which produces the classification decision (cover or stego).

Similarly to Xu-Net, the convolutional module has five blocks marked as 'Block 1' through 'Block 5' to extract effective features for cover and stego images discrimination; see Fig. 1. Each block is made of the following steps:

1. a **Convolution Layer**. Similar to Xu-Net [5], we set the size of the convolutional kernels to \(5 \times 5\) for Blocks 1 and 2, but we reduced it to \(3 \times 3\) for the Blocks 3 through 5. For all the convolution layers and similarly to ResNet [9] and Xu-Net [5], no biases are used. Biases terms are set to false on the convolution layer and moved to the Scale Layer.

2. an **Absolute Value** activation (ABS) layer. This ABS layer is only used in Block 1 similarly to Xu-Net. It forces the statistical modeling to consider the sign symmetry of the noise residuals. The relevance of this layer was observed in Xu-Net [5].

3. a **Batch Normalization (BN)**. The BN normalizes the distribution of each feature to a zero-mean and a unit-variance, and eventually, scales and translates the distribution. The benefit of using a BN layer is that it desensitizes the training to the parameters initialization [18], allows the use of a larger learning rate which speeds up the learning, and improves the detection accuracy [7]. Note that similarly to ResNet [9], and in contrast to Xu-Net, we provide a BN layer accompanied by a scale layer. The latter attempts to learn the scaling and translation parameters more efficiently. Those two parameters can be well learned by the independent Scale Layer. Similarly to ResNet, we observe a very slight increase in the network’s accuracy.

4. a non-linear **Activation layer**. For the Blocks 1 and 2, a **Truncation function** is used to limit the range of data values and prevent the deeper layers from modeling large values. Indeed, these values are sparse and not statistically significant. The formula of the truncation function \((\text{Trunc})\) is given in Eq. 1, and is parameterized by \(T \in \mathbb{N}\), a threshold:

\[
\text{Trunc}(x) = \begin{cases} 
-T, & x < -T, \\
-x, & -T \leq x \leq T, \\
T, & x > T.
\end{cases}
\]  

5. An **Average pooling**. This average pooling layer is exclusively used in Blocks 2 through 5. This allows to down-sample the feature maps, and thus reduces the dimensionality. For the last block, a **global average pooling** is performed to generate a one by one element for each corresponding feature map, thereby preventing the statistical modeling from grasping the location information of embedded pixels from the training data [19]. There is no pooling in the first block to avoid information loss at the beginning of the network.

The features extracted from the convolutional module feed the classification module which consists of three fully connected layers. The number of neurons in the first and second layers is 256 and 1024 respectively, and the last fully connected layer has only two neurons corresponding to the number of classes of the network’s output. At the end of this module, a softmax activation function is used to produce a distribution over the two class labels.

### 3. EXPERIMENTS

#### 3.1. Dataset and software platform

We use S-UNIWARD [20], and WOW [21], two well-known content-adaptive methods for the embedding in the spatial domain and their Matlab implementations (online codes\(^2\)) with the simulator for the embedding and a random key for each embedding. We thus avoid any wrong use of the C++ codes, i.e. a fixed and unique embedding key, as reported in [3].

Our steganalysis CNN, **Yedroudj-Net**, is compared with the state-of-the-art approaches: **Xu-Net** [5], **Ye-Net** [17], and with **SRM + EC** which stands for the hand-crafted feature set Spatial-Rich-Model [12] and the Ensemble Classifier [11]. For a fair comparison, all the involved steganalysis methods are tested on the same subsampled images from the BOSSBase database v.1.01 [22]. All CNNs experiments were performed with the publicly available Caffe toolbox [23] with necessary modifications, plus digits V5. All tests were run on an NVidia Titan X GPU card.

#### 3.2. Training, Validation, Test

Due to our GPU computing platform and time limitation, we conduct all the experiments on images of \(256 \times 256\) pixels, similarly to [17]. To this end, we resampled all the \(512 \times 512\) images to \(256 \times 256\) images, using the `imresize()` Matlab function with the default parameters. Then, our \(256 \times 256\) BOSSBase is split into two sets, 50% (resp. the other 50%) of the cover/stego pairs is assigned to the training.

\(^2\)http://dde.binghamton.edu/download/
During the CNNs training, we fixed a maximum of 900 epochs. Nevertheless, most of the time, we manually stopped the training when an over-fitting phenomenon appeared (usually before the epoch 200 for WOW and 300 for S-UNIWARD), i.e. when the Loss continues to decrease on the training set but starts to increase on the validation set. In practice, observing the Loss curve computed on the validation test leads us to keep two versions of the CNN: the CNN’s models with minimum Loss (resp. maximum) on the validation set over the previous five epochs. Those two CNN’s models are evaluated on the testing set, and we report the average error probability of detection for these two CNN’s models.

For SRM + EC we use the SRM feature set of dimension=34671 [12], and the Ensemble Classifier [11]. We report the minimum error probability under equals prior, averaged over 10 tests.

### 3.3. Hyper-parameters

We apply a mini-batch stochastic gradient descent (SGD) to train our CNN. The momentum is fixed to 0.95 and the weight decay to 0.0001. No dropout is used. The batch size in the training procedure is set to 16, due to GPU memory limitation (8 cover/stego pairs). All layers are initialized using Xavier method: the weights follow a Gaussian distribution and are chosen so that the variance for both input and output among each layer remains the same [24]. During the training, we use the step policy of Caffe to adjust the learning rate (initialized to 0.01). With this policy, each 10% of the total number of epochs, our learning rate is decreased by a factor gamma equal to 0.1. The threshold $T$, for the Truncations functions (see Equ. 1) is set to 3 for the first layer and 2 for the second layer, and the 30-basic high-pass SRM filters are not normalized. Note that the source codes and the materials files are available at http://www.lirmm.fr/~chaumont/DemoAndSources.html.

### 3.4. Difference between the 3 CNNs

In this section we will briefly discuss the differences between our CNN Yedroudj-Net, the Xu-Net CNN and Ye-Net CNN, the state-of-the-art CNNs for the spatial steganalysis. In our comparisons, Xu-Net is a CNN similar to the one given in [5] that takes images of size of 256×256 instead of 512×512. We thus suppressed the average pooling from the first Block, which is a favorable measure since it avoids an early down-sampling. We also set a ReLU activation function among the Fully connected layers. Fig.2 shows the overall architectures of all CNNs. We summarize below the major similarities and differences between the CNNs:

- Both Yedroudj-Net and Xu-Net use 5 convolution layers. Yedroudj-Net has nevertheless two times more features (256) at the input of the fully connected section. Ye-Net has more convolution layers.
- Both Yedroudj-Net and Xu-Net use a Batch Normalization layer; the Ye-Net does not.
- Both Yedroudj-Net and Xu-Net use the Absolute Value layer; the Ye-Net does not.
- Both Yedroudj-Net and Ye-Net use a 30 filter bank for pre-processing; the Xu-Net does not.
- Both Yedroudj-Net and Ye-Net use a Truncation activation function in Block 1 and 2 (We have found "Experimentally" that using Truncation activation function only in the Blocks 1 and 2 is the best choice in term of detection accuracy, those experiments are not reported here); the Xu-Net does not.
- Yedroudj-Net has three (resp. Xu-Net two, and Ye-Net one) fully connected layer.

### 3.5. Results without using any tricks

#### 3.5.1. General performance comparisons

In Table 1, we report the error probability obtained when steganalyzing WOW and S-UNIWARD embedding algorithms at 0.2 bpp and 0.4 bpp. The steganalysis methods are Yedroudj-Net, Xu-Net, Ye-Net, and SRM+EC [11, 12].

<table>
<thead>
<tr>
<th>Steganalysis</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2 bpp</td>
</tr>
<tr>
<td>SRM+EC [11, 12]</td>
<td>36.5 %</td>
</tr>
<tr>
<td>Yedroudj-Net</td>
<td>27.8 %</td>
</tr>
<tr>
<td>Xu-Net [5]</td>
<td>32.4 %</td>
</tr>
<tr>
<td>Ye-Net [17]</td>
<td>33.1 %</td>
</tr>
</tbody>
</table>
For WOW algorithm, Yedroudj-Net has an error probability 8% lower (resp. 11%) at 0.2 bpp (resp. 0.4 bpp) compared to SRM+EC. The results are also favorable for S-UNIWARD steganalysis with an equal error probability at 0.2 bpp and 2% lower at 0.4 bpp.

Compared to the other CNN algorithms, our proposed CNN achieves far superior results. Yedroudj-Net is 2% to 6% better compared to Xu-Net for the two embedding algorithms and the two payloads. The results are even better when compared to Ye-Net, where Yedroudj-Net is 3% to 9% better. Let us note that the two other CNNs are not always superior when compared to the SRM+EC. To beat SRM+EC, those approaches require using an ensemble of CNN, as proposed in [4], or increasing the learning database, as proposed in [6], and showed in section below.

Note that extreme caution must be taken for the initialization of the learning rate of the Ye-Net and the management of its evolution through the epochs. Indeed, a bad initialization prevents the network from converging. In Yedroudj-Net and Xu-Net, the use of the Batch Normalization ensures less sensitivity to such a parameter setting.

To conclude on these general comparisons, in a classical clairvoyant scenario without any channel-awareness, and without using an ensemble, a larger database, a virtual augmentation of the database, or a transfer learning, Yedroudj-Net has a clear advantage over all the state-of-the-art methods.

### 3.6. Results with a Base augmentation

Many tricks exist for improving the results of CNN but the base augmentation seems to be a very important measure to apply in order to better exploit the capacity of Deep Learning approaches.

Table 2: Base Augmentation influence: error probability comparison of Yedroudj, Xu and Ye nets on WOW at 0.2 bpp with a learning base augmented with BOWS2, and Virtually Augmented.

<table>
<thead>
<tr>
<th></th>
<th>BOSS</th>
<th>BOSS+BOWS2</th>
<th>BOSS+BOWS2+VA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yedroudj-Net</td>
<td>27.8%</td>
<td>23.7%</td>
<td>20.8%</td>
</tr>
<tr>
<td>Ye-Net</td>
<td>33.1%</td>
<td>26.1%</td>
<td>22.2%</td>
</tr>
<tr>
<td>Xu-Net</td>
<td>32.4%</td>
<td>30.3%</td>
<td>30.5%</td>
</tr>
</tbody>
</table>

In machine learning, and this is also true for CNNs, it is important to use a training base large enough to ensure a good generalization but also to avoid over-training. Some authors are prone to use big databases [2, 6, 17] in order to reach the state-of-the-art results. In the above experiment, we attempt to investigate the improvement brought by increasing the learning database size without modifying the testing set. It means that the learning set does not only contain images of the same kind as in the test set: e.g. the settings of cameras, the scenes of the learning set, can all be different from those of the testing set. We show the effects of increasing the image database on the error probability in Table 2. To increase the size of our training set, two scenarios have been tested inspired by [17].

In the first scenario, noted BOSS+BOWS2, we embedded the payload in the subsampled BOSSBase database v.1.01 [22]. We split this base into two sets: 50% of the cover/stego pairs to the training set, the rest to the testing set. Then, 10 000 additional pairs of cover/stego pair (obtained by subsampling BOWS2Base [25]) were added to the training set. The learning database now contains 15 000 pairs of cover/stego images minus 1000 pairs from BOSS, set aside for validation.

In the second scenario, noted BOSS+BOWS2+VA, the database is virtually augmented by performing the label-preserving flips and rotations on the BOSS+BOWS2 training set. The size of the BOSS+BOWS2 training set is thus increased by a factor of 8, which virtually gives a final learning database made of 112 000 pairs of cover/stego images plus 1000 pairs from BOSS used for validation.

Table 2 shows the performance comparisons in terms of detection error probability for Yedroudj-Net, Xu-Net [5], Ye-Net [17], against the embedding algorithm WOW [21] at payload 0.2 bpp. For all algorithms, better performances are achieved using BOSS+BOWS2 compared to using only BOSSBase. The Yedroudj-Net obtains the best results and decreases its detection error probability by 4%. Ye-Net and Xu-Net respectively decrease their detection error probability by 7% and 2%. At this point, it was not clear if the improvement was only due to a lack of data or also because the additional images came from the same cameras. We have nevertheless conducted additional experiments, reported in the paper [26], and it seems that in order to improve the performance, one must increase the database with images coming from the same sources and with a development process respecting the pixels resolutions and ratios.

When virtually augmenting the entire BOSS+BOWS2 learning set (i.e. BOSS+BOWS2+VA) thanks to the 8 combinations of rotations and flips that do not introduce interpolation, the performances are again increased. The Yedroudj-Net keeps the best results and decreases its detection error probability by 7% (Ye-Net decreases it by 11%, and Xu-Net by 2%) compared to the case of only using BOSSBase for the training. Comparing to RM+EC [11, 12], whose error probability is 36.5% with a learning on the BOSSBase, the Yedroudj-Net obtain an error probability of 20.8% which give an improvement of 16%. The Ye-Net obtains an improvement of 14% and the Xu-Net an improvement of 6%.

These tests reveal how important it is to have a large database when using CNN of 5-7 blocks. The number of parameters (without taking into account the BN and/or scale) goes approximately from 50 thousand (Xu-Net) to 500 thousand (Yedroudj-Net). Such a huge number of unknown requires bearing enough samples. The experiments show that the CNNs still do not have enough learning samples. For a steganalysis of BOSSBase with CNNs of 5-7 blocks, even 112 000 pairs of images (BOSS+BOWS2 virtually augmented) is not enough. Consequently, using a bigger base allows our CNN to achieve better performances even if the convergence time increases.

Using a GPU card of the previous generation (Nvidia TitanX) on an Intel Core i7-5930K CPU 3.50GHz × 12 with 32G of RAM, it takes less than one day for learning Yedroudj-Net CNN on BOSS-Base, three days on BOSS+BOWS2, and more than seven days on BOSS+BOWS2+VA.

### 4. CONCLUSION

This article presents the evaluation of the Yedroudj-Net CNN, designed for spatial steganalysis. This CNN gathers some recent design propositions in order to build a simple approach beating the state-of-the-art approaches in a classical clairvoyant scenario without knowledge of the selection channel.

The key to the steganalysis performance improvement is the combination of the following elements: a bank of filters for the preprocessing step, a Truncation activation function, and a Batch Normalization associated with a Scale Layer.

An additional experiment dealing with the problem of the learning base size showed that by adding BOWS2 and virtually augmenting the learning database, the results become extremely satisfactory. An experiment on WOW at 0.2 bpp led to an error probability decrease of 16% compared to the RM+EC.
5. REFERENCES


