Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch

Lionel PIBRE [2,3], Jérôme PASQUET [2,3], Dino IENCO [2,3],
Marc CHAUMONT [1,2,3]
(1) University of Nîmes, France
(2) University Montpellier, France
(3) CNRS, Montpellier, France

March 3, 2016

Media Watermarking, Security, and Forensics,

IS&T Int. Symp. on Electronic Imaging, SF, California, USA, 14-18 Feb. 2016.

The big promise of CNN...

Superlatives: lots of enthusiasm, fresh ideas, amazing results,...

	RM+EC	CNN	FNN
Max	24.93%	7.94%	8.92%
Min	24.21%	7.01%	8.44%
Variance	0.14	0.12	0.16
Average	24.67%	7.4%	8.66%

Table: Steganalysis results (P_E) with S-UNIWARD, 0.4 bpp, clairvoyant scenario, for RM+EC, CNN, and FNN

But ... experimental setup was artificial...

Outline

An example of Convolution Neural Network

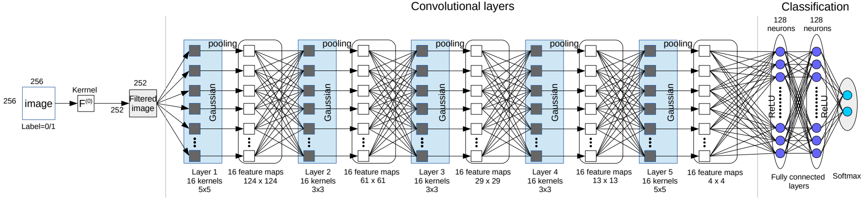


Figure: Qian *et al.* Convolutional Neural Network.

- Inspired from Krizhevsky *et al.* 2012 Network,
- Detection percentage only 3% to 4% lower than EC + RM.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, " ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems 25, NIPS'2012.

Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan, "Deep Learning for Steganalysis via Convolutional Neural Networks," in Proceedings of SPIE Media Watermarking, Security, and Forensics 2015.

Convolution Neural Network: Preliminary filter

$$F^{(0)} = \frac{1}{12} \begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix}$$

CNNs converge much slower without this preliminary high-pass filtering.

Convolution Neural Network: Layers

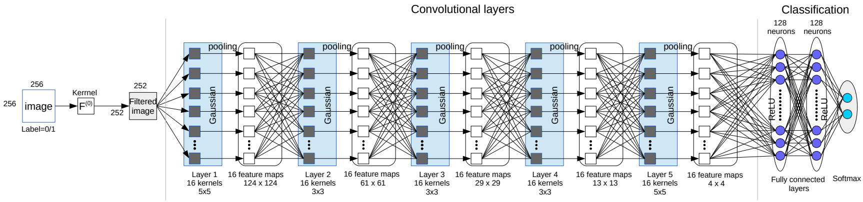


Figure: Qian *et al.* Convolutional Neural Network.

Inside one layer; successive steps:

- a convolution step,
- the application of an activation function,
- a pooling step,
- a normalization step.

Convolution Neural Network: Convolutions

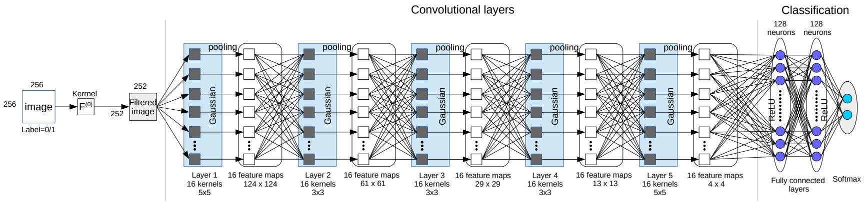


Figure: Qian *et al.* Convolutional Neural Network.

- First layer:

$$\tilde{I}_k^{(1)} = I^{(0)} \star F_k^{(1)}. \tag{1}$$

- Other layers:

$$\tilde{I}_k^{(l)} = \sum_{i=1}^{i=K^{(l-1)}} I_i^{(l-1)} \star F_{k,i}^{(l)}, \tag{2}$$

Convolution Neural Network: Activation

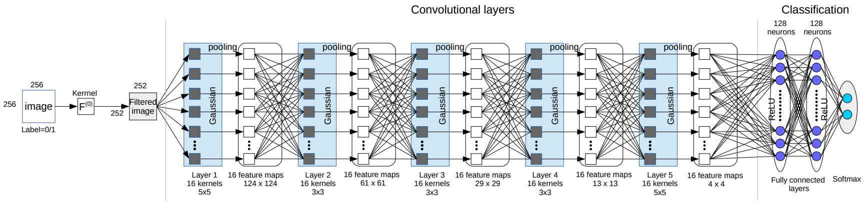


Figure: Qian *et al.* Convolutional Neural Network.

Possible activation functions:

- absolute function $f(x) = |x|$,
- sine function $f(x) = sinus(x)$,
- Gaussian function as in the Qian *et al.* network $f(x) = \frac{e^{-x^2}}{\sigma^2}$,
- ReLU (for Rectified Linear Units): $f(x) = max(0, x)$ as in our work,
- Hyperbolic tangent: $f(x) = tanh(x)$

Convolution Neural Network: Pooling

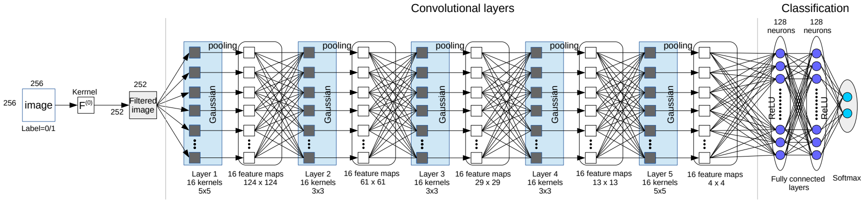


Figure: Qian *et al.* Convolutional Neural Network.

Pooling is a local operation computed on a neighborhood:

- local average (preserve the signal),
- or local maximum (translation invariance).

+ a sub-sampling operation.

For our "artificial" experiments, the pooling was not necessary.

Convolution Neural Network: Normalization

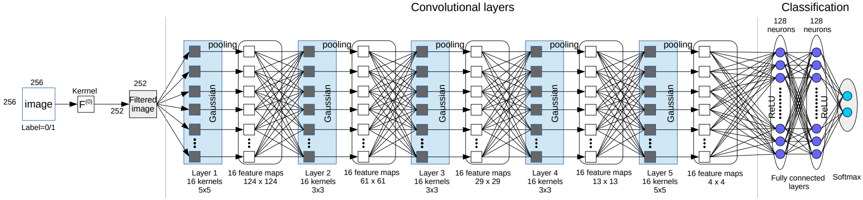


Figure: Qian *et al.* Convolutional Neural Network.

Case where normalization is done across the maps:

$$norm(I_k^{(1)}(x,y)) = \frac{I_k^{(1)}(x,y)}{\left(1 + \frac{\alpha}{size} \sum_{k'=max(0,k-\lfloor size/2 \rfloor)}^{k'=min(K,k-\lfloor size/2 \rfloor+size)} (I_{k'}^{(1)}(x,y))^2\right)^{\beta}}$$

Convolution Neural Network: Fully Connected Network

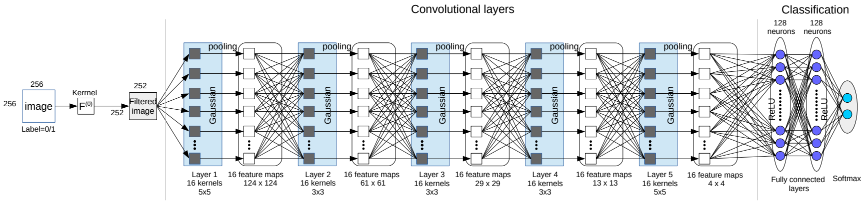


Figure: Qian *et al.* Convolutional Neural Network.

- three layers.
- a softmax function normalizes values between $[0, 1]$.
- the network delivers a value for cover (resp. for stego).

Our CNN

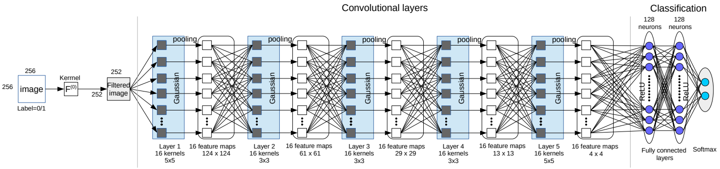


Figure: Qian *et al.* Convolutional Neural Network.

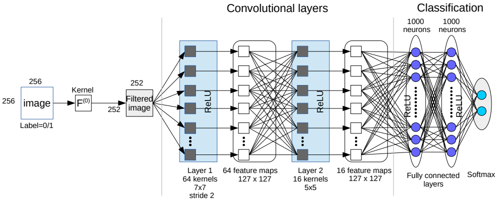
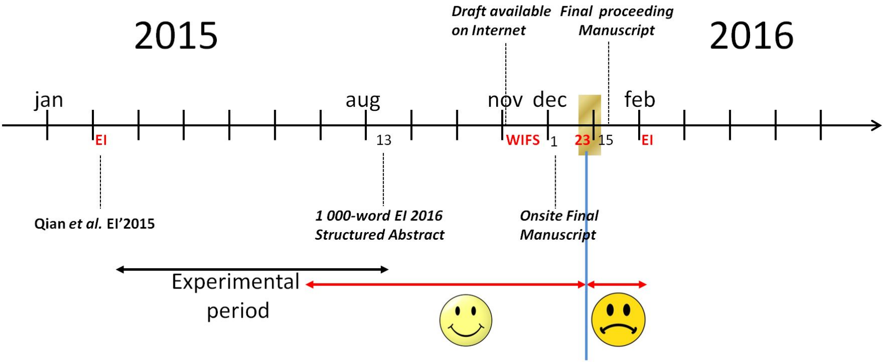


Figure: Our Convolutional Neural Network.

Outline

The story of that paper...



Outline

Experiences

- 40 000 images of size 256$\times$256 from BOSSBase,
- S-UNIWARD at 0.4 bits per pixels,
- Same embedding key and use of the simulator,
- learning on 60 000 images,

Why using the same key ?

We did not want to do that...

- Documentation error in the C++ S-UNIWARD software,
- Qian *et al.* 2015 have also misled.

$\Rightarrow$ We discovered this key problem the 23th of December 2015...

About the simulator

Probabilities for modifying a pixel x_i with $i \in \{1...n\}$ are:

- $p_i^{(-)} = \frac{exp(-\lambda \rho_i^{(-)})}{Z}$, for a -1 modification,
- $p_i^{(0)} = \frac{exp(-\lambda \rho_i^{(0)})}{Z}$, for no modification,
- $p_i^{(+)} = \frac{exp(-\lambda \rho_i^{(+)})}{Z}$, for a $+1$ modification,

where

- $\{\rho_i^{(-)}\}$, $\{\rho_i^{(0)}\}$, and $\{\rho_i^{(+)}\}$ are the changing costs,
- λ is obtained in order to respect the payload constraint,
- $Z = exp(-\lambda \rho_i^{(-)}) + exp(-\lambda \rho_i^{(0)}) + exp(-\lambda \rho_i^{(+)})$.

Using the same key...

Probabilities for modifying pixel x_i: $p_i^{(-)} = \frac{exp(-\lambda\rho_i^{(-)})}{Z}$, $p_i^{(0)} = \frac{exp(-\lambda\rho_i^{(0)})}{Z}$, and $p_i^{(+)} = \frac{exp(-\lambda\rho_i^{(+)})}{Z}$.

What happen when using the same key...

- The embedding key initialize the Pseudo-Random-Number-Sequence **Generator**,
- Whatever the image, the Pseudo Random Number Sequence $\in [0,1]^n$ is the same,
- The sequence is used to sample the distribution (see probabilities),
- Whatever the image, some position i will be most of the time always modified, and **always with the same "polarity" (-1 or +1)**...

This situation is artificial !!!

Illustration on the cropped BOSSBase database.

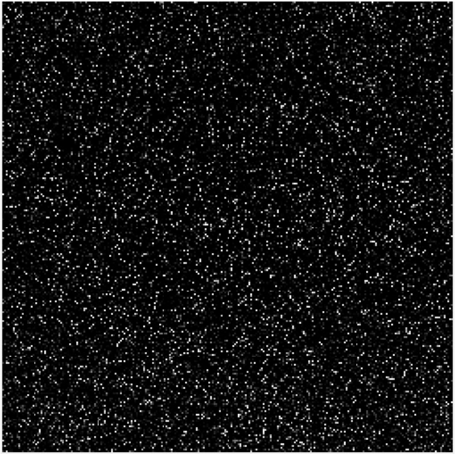


Figure: Probability of change. In white the most probable sites and in black the less probable ones.

Our best CNN in that "artificial scenario"

- 40 000 images of size 256×256 from BOSSBase,
- S-UNIWARD at 0.4 bits,
- Same embedding key and use of the simulator,
- learning on 60 000 images,

	RM+EC	CNN	FNN
Max	24.93%	7.94%	8.92%
Min	24.21%	7.01%	8.44%
Variance	0.14	0.12	0.16
Average	24.67%	7.4%	8.66%

Table: Steganalysis results (P_E) with S-UNIWARD, 0.4 bpp, clairvoyant scenario, for RM+EC, CNN, and FNN

But ... experimental setup was artificial...

Note that with different embedding keys, the same CNN structure in 2 layers, and with more neurons, the probability of error is 38.1%... There is still hope!

Conclusion on that story

- Be careful to the software's implementations!
- Be careful to use different keys for embedding!
- Be careful: the simulator only does a simulation (different from STC),
- Rich Models are under-efficient to detect the spatial phenomenons,

You will also find in the paper:

- Explanation/discussion on CNN, behavior of a CNN,
- A discussion on embedding keys,
- The presentation of the LIRMMBase.

End of talk

CNN is not dead...

... there is still things to do...

About LIRMMBase

LIRMMBase

" LIRMMBase: A database built from a mix of Columbia, Dresden, Photex, and Raise databases, and whose images do not come from the same cameras as the BOSSBase database. ",
L. Pibre, J. Pasquet, D. Ienco, and M. Chaumont,
LIRMM Laboratory, Montpellier, France, June 2015,
Website: www.lirmm.fr/ chaumont/LIRMMBase.html.