# Yedroudj-Net : un réseaux de neurones efficace pour la stéganalyse spatiale

M. Yedroudj[1] M. Chaumont[1,2] F. Comby[1]

[1] UNIVERSITE MONTPELLIER, UMR5506-LIRMM, F-34095 Montpellier Cedex 5, France

[2] UNIVERSITE DE NIMES, F-30021 Nîmes Cedex 1, France

{yedroudj, chaumont, comby}@lirmm.fr

## Résumé

*Pendant environ 10 ans, l'approche classique pour détecter la présence d'un message secret inséré dans une image était d'utiliser un ensemble de classifieurs alimentés par des vecteurs de caractéristiques issues des images à traiter. Ces dernières années, des études telles que Xu et al. ont indiqué que des réseaux de neurones convolutionnels (CNN) bien conçus peuvent atteindre des performances comparables aux approches classiques d'apprentissage automatique.*

*Dans cet article, nous proposons un CNN qui dépasse les performances de l'état de l'art en terme de probabilité d'erreur de classification. La proposition s'inscrit dans la continuité de ce qui a été proposé récemment et consiste en une fusion intelligente de briques importantes proposées dans divers articles. Parmi les éléments essentiels du CNN proposé, on peut citer l'utilisation : d'un ensemble de filtres pour le prétraitement de l'image d'entrée, de la troncature comme fonction d'activation, d'au moins cinq couches convolutionnelles avec une normalisation par lot (Batch normalization layer) et une couche de mise à l'échelle (Scale layer) ainsi qu'une couche entièrement connectée correctement dimensionnée.*

## Mots clefs

Stéganographie, stéganalyse, réseau de neurones convolutionnel, classification.

# 1 Introduction

Les premières tentatives d'utilisation des méthodes de Deep Learning pour la stéganalyse remontent à 2014 avec l'utilisation d'auto-encodeurs [1]. Un an plus tard, Qian *et al.* [2] et Pibre *et al.* [3] proposent d'utiliser des CNNs pour la stéganalyse. En 2016, les premiers résultats, similaires à ceux de l'état de l'art actuel, sont obtenus avec un ensemble de CNNs [4]. Le CNN Xu-Net [5] y est alors utilisé comme brique élémentaire.

D'autres réseaux ont été proposés en 2017 pour la stéganalyse des images JPEG. Dans l'article [6], les auteurs proposent d'utiliser un pré-traitement inspiré des Spatial Rich Models (SRM) ainsi qu'une grande base de données d'apprentissage. Les résultats obtenus sont alors proches de ceux de l'état de l'art. Dans l'approche décrite dans [7], le réseau est construit en tenant compte du découpage en bloc de $8\times8$ dû à la compression JPEG (notion de phase). Un ensemble de CNNs est requis pour obtenir des résultats légèrement meilleurs à ceux de l'état de l'art. Dans [8], un CNN inspiré de ResNet [9], avec un système de "shortcut connection" et vingt couches, permet également d'obtenir de meilleurs résultats que ceux de l'état de l'art.

Ces résultats sont très encourageants. Toutefois, si l'on compare aux améliorations obtenues dans d'autres domaines du traitement des images utilisant le Deep Learning [10], les résultats pour la stéganalyse ne sont pas "10%" meilleurs que ceux obtenus en utilisant un ensemble de classifieurs [11] avec un SRM [12, 13] ou un SRM avec connaissance du canal de sélection [14, 15]. En 2017, les principales voies explorées pour améliorer les résultats des CNNs sont : l'utilisation d'un ensemble de CNNs et la modification de la topologie en imitant le procédé d'extraction des SRM. Dans la plupart des cas, l'effort architectural (structure du réseau) ou expérimental (taille de la base) est très élevé pour une faible amélioration des performances.

En revenant sur les bonnes pratiques du Deep Learning et les études récentes, nous avons cherché à construire, expérimentalement, un réseau plus efficace que ceux de l'état de l'art, robuste au type d'images (non compressée ou compressée JPEG...) et sans nécessité d'utiliser un ensemble de CNNs (qui est connu pour améliorer les résultats mais au prix d'une complexité accrue). Dans cet article, nous présentons un CNN conçu pour la stéganalyse dans le domaine spatial qui permet d'obtenir de bons résultats sans avoir recours à une ou plusieurs astuces pour améliorer ses performances tel que : l'apprentissage par transfert [16], l'augmentation virtuelle de la taille de la base de données [17], etc. De plus, le réseau proposé est peu sensible à l'initialisation des hyper-paramètres et converge facilement (voir Section 4). Nous nommerons ce réseau "Yedroudj-Net CNN" et comparerons ses performances avec celles de deux autres réseaux : Xu-Net [5] et Ye-Net [17], et également avec un ensemble de classifieurs [12] combiné à un SRM [11] pour la stéganalyse dans le domaine spatial.

## 2 Stéganographie et stéganalyse

La stéganographie est l'art de dissimuler des informations au sein d'un support de sorte que ces informations soient indétectables pour un observateur. De nos jours, les méthodes stéganographiques les plus sûres **s'adaptent au contenu**. La plupart des algorithmes intègrent les données secrètes dans les régions avec un contenu complexe où les zones d'insertion sont moins détectables. Parmi les méthodes les plus performantes d'insertion dans le domaine spatial on trouve : WOW [18] et S-UNIWARD [19].

La stéganalyse, quant à elle, est l'art qui permet de détecter et d'extraire des données dissimulées au sein d'un support. La stéganalyse peut également servir de moyen efficace pour juger des performances de sécurité des techniques de stéganographie. En d'autres termes, une bonne méthode stéganographique doit être imperceptible non seulement pour le système de vision humain, mais indétectable aussi par analyses statistiques.

Jusqu'à présent, l'état de l'art de la stéganalyse utilise l'apprentissage automatique en appliquant deux étapes :

— l'extraction de vecteurs de caractéristiques pertinents à partir des images en utilisant un SRM [12],
— et l'utilisation d'un classifieur qui, à partir des vecteurs de caractéristiques, apprend un modèle permettant de distinguer les images modifiées (stégos) des images initiales (covers) [11].

## 3 Réseau de neurones convolutifs (CNN)

Un réseau neuronal est un modèle mathématique dont la conception s'inspire du fonctionnement des neurones biologiques. Il est composé de trois parties appelées couches qui sont elles-mêmes exclusivement composées de neurones. Un neurone effectue un produit scalaire entre les valeurs en entrée et ses paramètres (poids) et applique ensuite une fonction d'activation au résultat. Un réseau est composé de :

— la première couche qui est la couche dans laquelle les données à analyser sont injectées,
— la partie intermédiaire qui est un ensemble de couches appelées couches cachées,
— la dernière couche qui est la couche de sortie. Dans le cas d'une classification, elle affecte un score d'appartenance pour chacun des classes du problème.

Dans un réseau neuronal convolutif, une couche se décompose en trois étapes consécutives :

— l'application de convolution(s),
— l'application d'une fonction d'activation,
— la mise en commun des données.

Les données en entrée sont des images appelées carte de caractéristiques. Une couche de convolution produit également une carte de caractéristique

## 4 Yedrouj-Net

La figure.1 illustre la structure de notre CNN. Le réseau est composé d'un bloc de **prétraitement**, un bloc de **convolution** comprenant cinq couches, et finalement un bloc **entièrement connecté** comprenant trois couches et une couche de perte **« softmax »**. Le réseau prend en entrée des images de taille 256×256 pixels et fournit une distribution de probabilité pour les deux classes de sortie (image stego ou non).

Le bloc de **prétraitement** a pour objectif de ne conserver que les résidus de bruits hautes fréquences présents dans l'image d'entrée. L'image ainsi traitée est ensuite passée au CNN. Plusieurs articles [2, 3] ont remarqué que sans cette couche préliminaire le réseau convergeait beaucoup plus lentement. Le but de cette couche est de supprimer le contenu de l'image, réduire la dynamique de l'image et ainsi augmenter le rapport signal à bruit entre le signal stégo de faible amplitude (s'il est présent) et le signal lié à l'image. De cette façon, le CNN apprend sur un signal plus compact et plus robuste.

En s'inspirant des avantages de la *diversité* présentés dans [17], nous n'utilisons pas un seul filtre dans l'étape de prétraitement (comme présenté dans [2, 3, 5]) mais la banque de 30 filtres passe-haut présentés dans [12] dans le Spatial Rich Model (SRM). De ce fait, nous obtenons en sortie 30 cartes de caractéristiques. Il faut noter que les poids constituant les noyaux des filtres de l'étape de prétraitement ne sont pas optimisés (appris) durant la phase d'entraînement du réseau. Ces filtres ont été définis de façon à ce qu'ils n'augmentent pas trop la complexité structurelle du réseau. Ainsi, chaque noyau a été étendu sur un support de 5×5 avec en partie centrale les valeurs des poids données dans [12] et le reste a été complété avec des « 0 ». Aucune normalisation des noyaux n'a été effectuée. Le reste de notre CNN peut être divisé deux parties : la partie convolutionnelle visant à transformer l'image d'entrée en un *vecteur caractéristique* et la partie classification qui détermine si l'image est stéganographiée ou non. Comme dans le réseau de Xu *et al.*[5] (Xu-Net), la partie convolutionnelle est composée de cinq blocs repérés par « Bloc 1-5 » dans la figure Fig. 1. Ces blocs extraient des caractéristiques pertinentes des images permettant de déterminer ultérieurement si l'image est stégo ou non. Chaque bloc est composé de toutes ou une partie des fonctionnalités suivantes :

1- **Une couche de convolution**. Comme dans [5] la taille des noyaux des filtres de convolution est fixée à 5×5 pour les blocs 1 et 2 et est ensuite réduite à 1× 1 pour les blocs 3 à 5. Comme dans les réseaux Res-Net [9] et Xu-Net [5], aucun biais n'est utilisé (le terme biais est positionné à faux dans la configuration du réseau). Ce terme est géré dans la couche de mise à l'échelle (voir plus loin)
2- **Une couche d'activation ABS** (uniquement pour le premier bloc comme dans [5]) qui impose au modèle statistique de prendre en compte les symé-

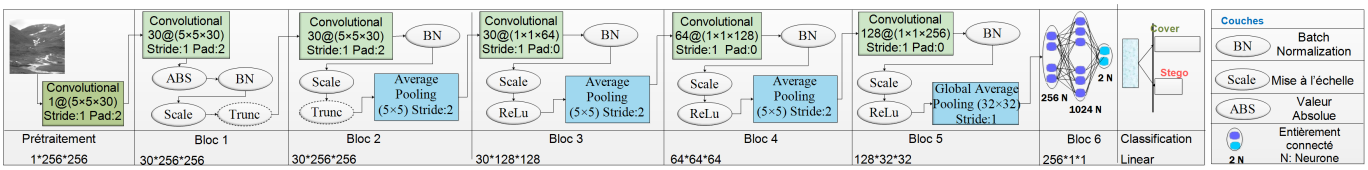


Figure 1 – *Yedrouj-Net CNN.*

tries de signe des résidus du bruit. L'utilité de cette couche a été montrée sur le réseau Xu-Net [5].

3- **Une étape de normalisation en lots (batch normalization)** qui vise à normaliser la distribution de chaque caractéristique afin de les rendre toutes comparables (voir Eq. 1). Cela transforme la distribution des caractéristiques en une distribution centrée (moyenne nulle) et de variance unitaire. Soit une variable aléatoire $X$ dont la réalisation est une caractéristique $x \in \mathbb{R}$, la normalisation en lot de $X$ est donné par l'équation Eq.1 :

$$BN(X, \gamma, \beta) = \beta + \gamma \frac{X - E[X]}{\sqrt{Var[X] + \epsilon}}, \quad (1)$$

avec $E[X]$ l'espérance mathématique de $X$, Var[X] sa variance et $\gamma$ et $\beta$ deux facteurs respectivement pour le décalage et la mise à l'échelle. Dans l'article original [20] les auteurs préconisaient de calculer $E[X]$ et Var[X] et d'apprendre $\gamma$ et $\beta$ avec le reste des paramètres du réseau. La normalisation rend l'apprentissage des paramètres moins sensible à l'initialisation [20], et permet d'utiliser un taux d'apprentissage plus grand, ce qui accélère la phase d'apprentissage et améliore la précision de détection [7]. Nos expériences ont montré que séparer la normalisation de la mise à l'échelle augmentait légèrement la précision du réseau. On imposera donc $\beta = 0$ et $\gamma = 1$ pour cette couche de batch normalisation.

4- **Une couche de mise à l'échelle**. Comme dans ResNet [9] et à la différence du Xu-Net [5] qui utilise la couche de normalisation pour apprendre les valeurs de $\gamma$ et $\beta$ , nous déportons cette capacité à la couche de mise à l'échelle. On aura donc la fonction de mise à l'échelle suivante :

$$Scale(X, \gamma, \beta) = \beta + \gamma X, \quad (2)$$

5- **Une couche d'activation non linéaire**. Pour les blocs 1 et 2, une fonction *troncature* est utilisée pour limiter la dynamique des valeurs et empêcher les couches plus profondes d'utiliser de grandes valeurs éparses et statistiquement non significatives. La fonction troncature $Trunc$ est donnée par l'équation 3 :

$$Trunc(x) = \begin{cases} -T, si & x < -T \\ x, si & -T \leq x \leq T \\ T, si & x > T \end{cases} \quad (3)$$

où $T \in \mathbb{N}$ est un seuil. Cette suppression des valeurs aberrantes, proposée dans [17], rend le processus plus robuste. Pour les blocs 3 à 5 la fonction d'activation ReLU (*Rectified Linear Unit*) a été utilisée car elle donne de bonnes performances et que le calcul de son gradient est rapide.

6- **Une couche de pooling moyenné**. Cette couche est seulement utilisée dans les couches 2 à 5. Elle permet de faire un sous-échantillonnage des cartes de caractéristiques et ainsi de réduire leur taille. On perd ainsi des informations spatiales mais on limite le risque de sur-apprentissage [21].

Les caractéristiques extraites du module convolutif sont transmises au module de classification qui se compose de trois couches entièrement connectées. Le nombre de neurones dans la première et la deuxième couche est de 256 et 1024 respectivement, et la dernière couche entièrement connectée n'a que deux neurones correspondant au nombre de classes de sortie du réseau. À la fin de ce module, une fonction d'activation softmax est utilisée pour retourner un score pour les deux classes cover et stego.

# 5 Expérimentations

Nous présentons dans cette section le contexte expérimental ainsi qu'une comparaison des performances de notre réseau avec 3 autres approches.

## 5.1 Base de données d'images et plate-formes logicielles

Pour l'insertion de données dans le domaine spatial nous avons utilisé deux méthodes très connues s'adaptant au contenu de l'image : S- UNIWARD [19] et WOW [18].

Pour la partie stéganalyse, nous avons comparé notre réseau *Y*edrouj-Net avec deux approches de l'état de l'art par CNN : les réseaux Xu-Net [5] et Ye-Net [17], et avec une méthode basée SRM [12] couplé avec un ensemble de classifieurs [11].

Pour que la comparaison soit significative, toutes les méthodes de stéganalyse ont été testées sur des images sous échantillonnées de la base de données d'images BOSS-Base v.1.01 [22] (voir la section 5.2). Les algorithmes d'insertion utilisent les codes en Matlab disponibles en ligne [1] avec un simulateur pour l'insertion et une clef d'insertion aléatoire pour chaque insertion. Nous évitons ainsi toute erreur d'utilisation des codes en C++, i.e. l'utilisation d'une clef unique d'insertion comme mentionné dans [3].

1. http ://dde.binghamton.edu/download/

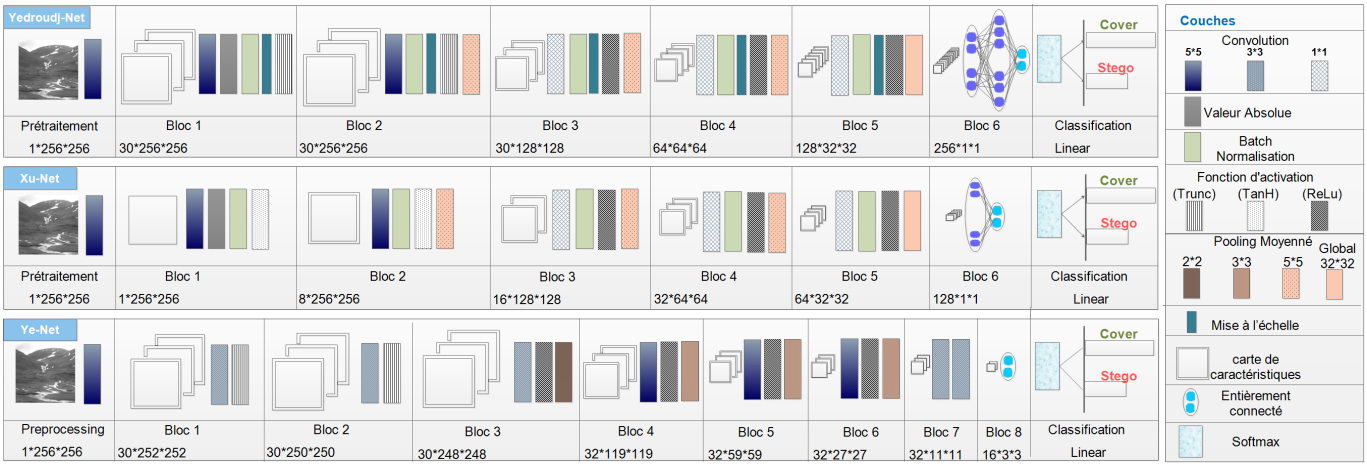


Figure 2 – *Comparison entre Yedroudj-Net, Xu-Net, et Ye-Net.*

Toutes les expérimentations avec les CNNs ont été réalisées en utilisant la toolbox **Caffe** [23] avec les modifications nécessaires et digits V5. La machine ayant servi pour ces tests est équipée d'une carte graphique NVidia Titan X.

## 5.2 Apprentissage, Validation et tests

Pour prendre en compte les limitations de notre carte graphique ainsi que limiter le temps de calcul, nous avons réalisé les expérimentations sur des images de taille 256×256 pixels (à l'instar de [17]). Pour ceci, toutes les images 512x512 ont été redimensionnées en utilisant la fonction *imresize()* de Matlab avec les paramètres par défaut (interpolation bicubique).

Notre base d'images issues de la BOSSBase est divisée en deux sous-ensembles : la moitié des paires cover/stego pour l'apprentissage et l'autre moitié pour le test. Seules 4000 paires d'images sur les 5000 disponibles de l'ensemble d'apprentissage sont sélectionnées aléatoirement pour l'apprentissage des poids du réseau ; les autres 1000 restantes sont utilisées pour la validation. Les images de l'ensemble de test ne sont jamais utilisées lors de l'entraînement du réseau.

Pour les CNNs, nous avons imposé le nombre maximum d'époque à 500 durant l'apprentissage. Néanmoins, la plupart du temps nous stoppions manuellement la convergence dès lors qu'un phénomène de sur-apprentissage apparaissait. Ceci se manifestait généralement par une valeur de fonction de perte (*Loss*) qui décroit sur l'ensemble d'apprentissage et au contraire qui augmente sur l'ensemble de validation. L'observation de l'évolution de la courbe de la fonction de perte calculée sur le sous-ensemble d'images pour la validation permet de conserver deux modèles du réseaux : celui dont la valeur de la fonction de perte est minimum (resp. maximum) sur les 5 dernières époques. Ces deux modèles de réseaux sont évalués sur l'ensemble d'images de test et on retourne la moyenne de la probabilité d'erreur de détection de ces deux modèles. Pour le SRM plus l'ensemble de classifieurs, nous avons utilisé l'ensemble de caractéristiques SRM de dimension 34671 [12] et l'ensemble de classifieurs de [11]. Nous donnons les résultats de la moyenne sur 10 tests de la probabilité d'erreur minimum, avec a priori égales.

## 5.3 Hyper-parameters

Afin d'entraîner notre CNN, nous appliquons une descente de gradient stochastique (SGD) sur les mini-batchs. Le momentum est fixé à 0.95 et le weight decay à 0.0001. il n'y a pas de dropout. La taille du mini-batch dans le processus d'apprentissage est initialisé à 16 soit 8 paires cover/stego. Toutes les couches sont initialisées en utilisant l'initialisation de "Xavier" qui initialise les poids de telle sorte que la variance de la distribution gaussienne de l'entrée et de la sortie soit identique [24].

Durant l'apprentissage, on utilise la stratégie itérative de caffe "step policy" pour ajuster le taux d'apprentissage qui est initialisé à 0.01. Une telle stratégie induit une réduction du taux d'apprentissage d'un facteur gamma (égal à 0.1) tout les 10% du nombre total d'époques. Les seuils T des fonctions de troncature valent respectivement 3 et 2 pour la première et la deuxième couche (voir Eq 3).

## 5.4 Résultats

Le Tableau 1, présente les probabilités d'erreurs obtenues en opérant une stéganalyse avec deux algorithmes d'insertion WOW et S-UNIWARD pour des payloads respectivement de 0.2 et 0.4 bits par pixels (bpp). Les quatre méthodes de stéganalyse comparées sont : le Yedroudj-Net, le Xu-Net [5], le Ye-Net [17] et le SRM+EC [11, 12]. Pour l'algorithme WOW, le Yedroudj-Net a des probabilités d'erreur plus basses de 8% et de 11% respectivement pour des payloads de 0.2 bpp et 0.4 bpp, comparé au SRM+EC. Les résultats restent bons pour la stéganalyse de S-UNIWARD avec une probabilité d'erreur égale pour un payload de 0.2 bpp et 2% plus faible pour un payload de 0.4 bpp.

Notre CNN présente de meilleurs résultats que les autres algorithmes de CNN. Le Yedroudj-Net est meilleur de 2%

Tableau 1 – *Comparaison de Yedroudj-Net et trois méthodes de stéganalyse état de l'art. Nous rapportons la probabilité d'erreur obtenue en utilisant les deux algorithmes d'insertion WOW [18] et S-UNIWARD [19], à 0,2 bpp et 0,4 bpp. Les méthodes de stéganalyse sont Yedroudj-Net, Xu-Net [5], Ye-Net [17] et SRM+EC [11, 12].*

| | *BOSS 256×256* | | | |
|---|---|---|---|---|
| | **WOW** | | **S-UNIWARD** | |
| | 0.2 bpp | 0.4 bpp | 0.2 bpp | 0.4 bpp |
| SRM+EC | 36.5 % | 25.5 % | **36.6** % | 24.7 % |
| Yedroudj-Net | **27.8** % | **14.1** % | **36.7** % | **22.8** % |
| Xu-Net | 32.4 % | 20.7 % | 39.1 % | 27.2 % |
| Ye-Net | 33.1 % | 23.2 % | 40.0 % | 31.2 % |

à 6% que le Xu-Net pour les deux algorithmes et les deux payloads. Ses performances sont de loin meilleures que celle du Ye-Net de 3% à 9%. Notons cependant que les deux autres CNNs ne sont pas plus performants que le SRM+EC. Afin d'être meilleurs que le SRM+EC, il est nécessaire qu'ils utilisent un ensemble de CNNs comme dans [4] ou qu'ils augmentent la base d'apprentissage [6]. L'initialisation du taux d'apprentissage du Ye-Net ainsi que la gestion de son évolution à travers les époques nécessitent beaucoup de minutie. En effet, une mauvaise initialisation empêche le réseau de converger. Dans le Yedroudj-Net et le Xu-Net, la normalisation en lot assure moins de sensibilité. En guise de conclusion à ces comparaisons générales, nous retenons que dans un scénario classique clairvoyant sans connaissance du canal de sélection et sans utiliser d'ensemble, de transfert d'apprentissage, de base de données plus grande ou virtuellement augmentée, le Yedroudj-Net a un avantage certain sur les autres méthodes de l'état de l'art.

# 6 Conclusion

Dans cet article, nous présentons et évaluons un nouveau CNN conçu pour la stéganalyse dans le domaine spatial, le Yedroudj-Net. Ce CNN est simple, et donne de meilleures performances que l'état de l'art dans un scénario classique clairvoyant sans connaissance du canal de sélection.
Les parties importantes de son architecture sont une banque de filtres pour l'étape de pré-traitement, la fonction d'activation de troncature et la normalisation en lots associée à la couche de mise à l'échelle.

# Références


[1] S. Tan et B. Li. Stacked convolutional auto-encoders for steganalysis of digital images. Dans *Proceedings of Signal and Information Processing Association Annual Summit and Conference, APSIPA'2014*, pages 1–4, Siem Reap, Cambodia, Décembre 2014.

[2] Yinlong Qian, Jing Dong, Wei Wang, et Tieniu Tan. Deep Learning for Steganalysis via Convolutional Neural Networks. Dans *Proceedings of Media Watermarking, Security, and Forensics 2015, MWSF'2015, Part of IS&T/SPIE Annual Symposium on Electronic Imaging, SPIE'2015*, volume 9409, pages 94090J–94090J–10, San Francisco, California, USA, Février 2015.

[3] L. Pibre, J. Pasquet, D. Ienco, et M. Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch. Dans *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2016, Part of I&ST International Symposium on Electronic Imaging, EI'2016*, pages 1–11, San Francisco, California, USA, Février 2016.

[4] Guanshuo Xu, Han-Zhou Wu, et Yun Q. Shi. Ensemble of CNNs for Steganalysis : An Empirical Study. Dans *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'16, pages 103–107, Vigo, Galicia, Spain, Juin 2016.

[5] G. Xu, H.-Z. Wu, et Y. Q. Shi. Structural Design of Convolutional Neural Networks for Steganalysis. *IEEE Signal Processing Letters*, 23(5) :708–712, Mai 2016.

[6] Jishen Zeng, Shunquan Tan, Bin Li, et Jiwu Huang. Pre-training via fitting deep neural network to rich-model features extraction procedure and its effect on deep learning for steganalysis. Dans *Proceedings of Media Watermarking, Security, and Forensics 2017, MWSF'2017, Part of IS&T Symposium on Electronic Imaging, EI'2017*, page 6, Burlingame, California, USA, Janvier 2017.

[7] Mo Chen, Vahid Sedighi, Mehdi Boroumand, et Jessica Fridrich. JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images. Dans *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'17, page 10, Drexel University in Philadelphia, PA, Juin 2017.

[8] Guanshuo Xu. Deep Convolutional Neural Network to Detect J-UNIWARD. Dans *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'17, page 6, Drexel University in Philadelphia, PA, Juin 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et Jian Sun. Deep residual learning for image recognition. Dans *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR'2016*, pages 770–778, Las Vegas, Nevada, Juin 2016.

[10] Yann LeCun, Yoshua Bengio, et Geoffrey Hinton. Deep learning. *Nature*, 521(7553) :436–444, Mai 2015.

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

[12] J. Fridrich et J. Kodovský. Rich Models for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics and Security, TIFS*, 7(3) :868–882, June 2012.

[13] Chao Xia, Qingxiao Guan, Xianfeng Zhao, Zhoujun Xu, et Yi Ma. Improving GFR Steganalysis Features by Using Gabor Symmetry and Weighted Histograms. Dans *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'17, page 11, Drexel University in Philadelphia, PA, Juin 2017.

[14] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, et J. Fridrich. Selection-channel-aware rich model for steganalysis of digital images. Dans *Proceedings of IEEE International Workshop on Information Forensics and Security, WIFS'2014*, pages 48–53, Décembre 2014.

[15] T. Denemark, M. Boroumand, et J. Fridrich. Steganalysis features for content-adaptive jpeg steganography. *IEEE Transactions on Information Forensics and Security*, 11(8) :1736–1746, Août 2016.

[16] Y. Qian, J. Dong, W. Wang, et T. Tan. Learning and transferring representations for image steganalysis using convolutional neural network. Dans *Proceedings of IEEE International Conference on Image Processing, ICIP'2016*, pages 2752–2756, Phoenix, Arizona, Septembre 2016.

[17] Jian Ye, Jiangqun Ni, et Yang Yi. Deep Learning Hierarchical Representations for Image Steganalysis. *Accepted to IEEE Transactions on Information Forensics and Security, TIFS*, page 13, 2017.

[18] V. Holub et J. Fridrich. Designing Steganographic Distortion Using Directional Filters. Dans *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS'2012*, pages 234–239, Tenerife, Spain, Décembre 2012.

[19] V. Holub, J. Fridrich, et T. Denemark. Universal Distortion Function for Steganography in an Arbitrary Domain. *EURASIP Journal on Information Security, JIS*, 2014(1), 2014.

[20] Sergey Ioffe et Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. Dans *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.

[21] Min Lin, Qiang Chen, et Shuicheng Yan. Network in network. *arXiv preprint arXiv :1312.4400*, 2013.

[22] P. Bas, T. Filler, et T. Pevný. 'Break Our Steganographic System' : The Ins and Outs of Organizing BOSS. Dans *Proceedings of the 13th International Conference on Information Hiding, IH'2011*, volume 6958 de *Lecture Notes in Computer Science*, pages 59–70, Prague, Czech Republic, Mai 2011. Springer.

[23] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, et Trevor Darrell. Caffe : Convolutional architecture for fast feature embedding. Dans *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[24] Xavier Glorot et Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. Dans *Aistats*, volume 9, pages 249–256, 2010.